cloudera®

Apache Hive Guide

Cloudera, Inc.
395 Page Mill Road
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: Cloudera Enterprise 5.9.x
Date: February 3, 2021

Table of Contents

Using Apache Hive with CDH

Hive data warehouse software enables reading, writing, and managing large datasets in distributed storage. Using the Hive query language (HiveQL), which is very similar to SQL, queries are converted into a series of jobs that execute on a Hadoop cluster through MapReduce or Apache Spark.

Users can run batch processing workloads with Hive while also analyzing the same data for interactive SQL or machine-learning workloads using tools like Apache Impala or Apache Spark—all within a single platform.

As part of CDH, Hive also benefits from:

- Unified resource management provided by YARN
- Simplified deployment and administration provided by Cloudera Manager
- Shared security and governance to meet compliance requirements provided by Apache Sentry and Cloudera Navigator

Continue reading:

Use Cases for Hive

Because Hive is a petabyte-scale data warehouse system built on the Hadoop platform, it is a good choice for environments experiencing phenomenal growth in data volume. The underlying MapReduce interface with HDFS is hard to program directly, but Hive provides an SQL interface, making it possible to use existing programming skills to perform data preparation.

Hive on MapReduce or Spark is best-suited for batch data preparation or ETL:

- You must run scheduled batch jobs with very large ETL sorts with joins to prepare data for Hadoop. Most data served to BI users in Impala is prepared by ETL developers using Hive.

- You run data transfer or conversion jobs that take many hours. With Hive, if a problem occurs partway through such a job, it recovers and continues.

- You receive or provide data in diverse formats, where the Hive SerDes and variety of UDFs make it convenient to ingest and convert the data. Typically, the final stage of the ETL process with Hive might be to a high-performance, widely supported format such as Parquet.

Installation

On a cluster managed by Cloudera Manager, Hive comes along with the base CDH installation and does not need to be installed separately. With Cloudera Manager, you can enable or disable the Hive service, but the Hive component always remains present on the cluster.

On an unmanaged cluster, you can install Hive manually, using packages or tarballs with the appropriate command for your operating system.

Install the appropriate Hive packages using the appropriate command for your distribution.

OS	Command
RHEL-compatible	`$ sudo yum install <pkg1> <pkg2> ...`
SLES	`$ sudo zypper install <pkg1> <pkg2> ...`
Ubuntu or Debian	`$ sudo apt-get install <pkg1> <pkg2> ...`

The packages are:

- `hive` – base package that provides the complete language and runtime
- `hive-metastore` – provides scripts for running the metastore as a standalone service (optional)
- `hive-server2` – provides scripts for running HiveServer2
- `hive-hbase` - optional; install this package if you want to use Hive with HBase.

See Installing Hive on page 17 for details about installing and configuring Hive components.

To access the Hive server with JDBC clients, such as Beeline, install the JDBC driver for HiveServer2 that is defined in `org.apache.hive.jdbc.HiveDriver`.

See Installing the Hive JDBC Driver on Clients on page 39 for details about installing the JDBC drivers and the connection URLs to use to connect to HiveServer2 from Hive clients.

Upgrading

Upgrade Hive on all the hosts on which it is running including both servers and clients.

See Upgrading Hive on page 13 for details about deprecated versions, upgrading best practices, and information about upgrading the Hive metastore schema.

Configuration

Hive offers a number of configuration settings related to performance, file layout and handling, and options to control SQL semantics. Depending on your cluster size and workloads, configure HiveServer2 memory, table locking behavior, and authentication for connections. See Configuring HiveServer2 on page 32 for details about required configuration changes that you must perform.

The Hive metastore service, which stores the metadata for Hive tables and partitions, must also be configured. See Configuring the Hive Metastore on page 19 for details about deployment modes, information about supported metastore databases, and specific configurations for MySQL, PostgreSQL, and Oracle.

To configure Hive to use the Amazon S3 filesystem for transient ETL jobs, see Configuring Transient Hive ETL Jobs to Use the Amazon S3 Filesystem on page 50

The Metastore Database

The metastore database is an important aspect of the Hive infrastructure. It is a separate database, relying on a traditional RDBMS such as MySQL or PostgreSQL, that holds metadata about Hive databases, tables, columns, partitions, and Hadoop-specific information such as the underlying data files and HDFS block locations.

The metastore database is shared by other components. For example, the same tables can be inserted into, queried, altered, and so on by both Hive and Impala. Although you might see references to the “Hive metastore”, be aware that the metastore database is used broadly across the Hadoop ecosystem, even in cases where you are not using Hive itself.

The metastore database is relatively compact, with fast-changing data. Backup, replication, and other kinds of management operations affect this database. See Configuring the Hive Metastore on page 19 for details about configuring the Hive metastore.

Cloudera recommends that you deploy the Hive metastore, which stores the metadata for Hive tables and partitions, in “remote mode.” In this mode the metastore service runs in its own JVM process and other services, such as HiveServer2, HCatalog, and Apache Impala communicate with the metastore using the Thrift network API.

See Starting the Metastore on page 34 for details about starting the Hive metastore service.

HiveServer2

HiveServer2 is a server interface that enables remote clients to submit queries to Hive and retrieve the results. It replaces HiveServer1, which has been deprecated and will be removed in a future release of CDH. HiveServer2 supports multi-client concurrency, capacity planning controls, Sentry authorization, Kerberos authentication, LDAP, SSL, and provides better support for JDBC and ODBC clients.

HiveServer2 is a container for the Hive execution engine. For each client connection, it creates a new execution context that serves Hive SQL requests from the client. It supports JDBC clients, such as the Beeline CLI, and ODBC clients. Clients connect to HiveServer2 through the Thrift API-based Hive service.

See Configuring HiveServer2 on page 32 for details on configuring HiveServer2 and see Starting, Stopping, and Using HiveServer2 on page 35 for details on starting/stopping the HiveServer2 service and information about using the Beeline CLI to connect to HiveServer2. For details about managing HiveServer2 with its native web user interface (UI), see HiveServer2 Web UI on page 45.

Hive on Spark

Hive traditionally uses MapReduce behind the scenes to parallelize the work, and perform the low-level steps of processing a SQL statement such as sorting and filtering. Hive can also use Spark as the underlying computation and parallelization engine. See Running Hive on Spark on page 43 for details about configuring Hive to use Spark as its execution engine and see Tuning Hive on Spark on page 64 for details about tuning Hive on Spark.

Hive and HBase

Apache HBase is a NoSQL database that supports real-time read/write access to large datasets in HDFS. See Using Hive with HBase on page 36 for details about configuring Hive to use HBase. For information about running Hive queries on a secure HBase server, see Using Hive to Run Queries on a Secure HBase Server on page 93.

Hive on Amazon S3

Use the Amazon S3 filesystem to efficiently manage transient Hive ETL (extract-transform-load) jobs. For step-by-step instructions to configure Hive to use S3 and multiple scripting examples, see Configuring Transient Hive ETL Jobs to

Use the Amazon S3 Filesystem. To optimize how Hive writes data to S3-backed tables and partitions, see Optimizing Hive Write Performance on Amazon S3.

Transaction (ACID) Support in Hive

Hive in CDH does not support transactions (HIVE-5317). Currently, transaction support in Hive is an experimental feature that only works with the ORC file format. Cloudera recommends using the Parquet file format, which works across many tools. Merge updates in Hive tables using existing functionality, including statements such as `INSERT`, `INSERT OVERWRITE`, and `CREATE TABLE AS SELECT`.

Managing Hive

Cloudera recommends using Cloudera Manager to manage Hive services, which are called managed deployments. If yours is not a managed deployment, configure HiveServer2 Web UI to manage Hive services.

Using Cloudera Manager to Manage Hive

Cloudera Manager uses the Hive metastore, HiveServer2, and the WebHCat roles to manage the Hive service across your cluster. Using Cloudera Manager, you can configure the Hive metastore, the execution engine (either MapReduce or Spark), and manage HiveServer2.

See Managing Hive Using Cloudera Manager on page 42

Using HiveServer2 Web UI to Manage Hive

The HiveServer2 web UI provides access to Hive configuration settings, local logs, metrics, and information about active sessions and queries. The HiveServer2 web UI is enabled in newly created clusters running CDH 5.7 and higher, and those using Kerberos are configured for SPNEGO. Clusters upgraded from a previous CDH version must be configured to enable the web UI; see HiveServer2 Web UI Configuration on page 45.

Ingesting Data with Hive

Hive can ingest data into several different file formats, such as Parquet, Avro, TEXTFILE, or RCFile. If you are setting up a data pipeline where Apache Impala is involved on the query side, use Parquet. If a custom file format is required, you can extend the Hive SerDes. See the Apache Hive wiki for information about the Hive SerDes and how to write your own for Hive.

Important:

The configuration property `serialization.null.format` is set in Hive and Impala engines as SerDes or table properties to specify how to serialize/deserialize NULL values into a storage format.

This configuration option is suitable for text file formats only. If used with binary storage formats such as RCFile or Parquet, the option causes compatibility, complexity and efficiency issues.

Column and Table Statistics for Query Optimization

Statistics for Hive can be numbers of rows of tables or partitions and the histograms of interesting columns. Statistics are used by the cost functions of the query optimizer to generate query plans for the purpose of query optimization.

See Hive Table Statistics on page 46 for details about collecting statistics for Hive.

Tuning

Tuning Hive consists of configuring numerous Hive parameters for better performance and scalability. The most important among these settings is configuring sufficient memory for HiveServer2 and the Hive metastore. This includes allocating memory for heap size based upon the number of concurrent connections that are typical for your deployment. Configuring garbage collection limits and keeping the number of table partitions below recommended limits are also important when tuning Hive performance. See Tuning Hive on page 54 for details about recommended limits and best practices. If you are using Spark as your execution engine, see Tuning Hive on Spark on page 64.

High Availability

Enable high availability for Hive by configuring a load balancer to manage HiveServer2 and by enabling high availability for the Hive metastore.

To enable high availability for multiple HiveServer2 hosts, configure a load balancer to manage them. To increase stability and security, configure the load balancer on a proxy server. The following sections describe how to enable high availability by using Cloudera Manager or how to enable it manually for unmanaged clusters.

See Configuring HiveServer2 High Availability in CDH on page 71 for details about configuring a load balancer for HiveServer2.

You can enable Hive metastore high availability (HA) so that your cluster is resilient to failures if a metastore becomes unavailable.

See Hive Metastore High Availability on page 68 for details about enabling the metastore for high availability.

Replication

Hive replication enables you to copy (replicate) your Hive metastore and data from one cluster to another and synchronize the Hive metastore and data set on the *destination* cluster with the source, based on a specified replication schedule. The destination cluster must be managed by the Cloudera Manager Server where the replication is being set up, and the *source* cluster can be managed by that same server or by a peer Cloudera Manager Server.

See Hive Replication on page 76 for details about using Cloudera Manager to set up data replication for Hive.

Security

Securing Hive involves configuring or enabling:

- **Authentication** for Hive Metastore, HiveServer2, and all Hive clients with your deployment of LDAP and Kerberos for your cluster.

  See Hive Authentication on page 86, HiveServer2 Security Configuration on page 86, Hive Metastore Server Security Configuration on page 92, and Using Hive to Run Queries on a Secure HBase Server on page 93 for details.

- **Authorization** for HiveServer2 using role-based, fine-grained authorization that is implemented with Apache Sentry policies. You must configure HiveServer2 authentication before you configure authorization because Apache Sentry depends on an underlying authentication framework to reliably identify the requesting user.

  See Sentry Policy File Authorization, User to Group Mapping, and Authorization Privilege Model for Hive and Impala for details. Configure Sentry permissions using GRANT and REVOKE statements using the HiveServer2 client, the Beeline CLI. See Hive SQL Syntax for Use with Sentry on page 98 for details.

> **Important:** Cloudera does not support Apache Ranger or Hive's native authorization frameworks for configuring access control in Hive. Use Cloudera-supported Apache Sentry instead.

- **Encryption** to secure the network connection between HiveServer2 and Hive clients.

  Starting with CDH 5.5, encryption for HiveServer2 clients has been decoupled from the authentication mechanism. This means you can use either SASL QOP or TLS/SSL to encrypt traffic between HiveServer2 and its clients, irrespective of whether Kerberos is being used for authentication. Previously, the JDBC client drivers only supported SASL QOP encryption on Kerberos-authenticated connections.

  SASL QOP encryption is better suited for encrypting RPC communication and may result in performance issues when dealing with large amounts of data. Move to using TLS/SSL encryption to avoid such issues.

  This topic describes how to set up encrypted communication between HiveServer2 and its JDBC/ODBC client drivers.

  See Configuring Encrypted Communication Between HiveServer2 and Client Drivers on page 95 for details.

Troubleshooting

See Troubleshooting Hive on page 105 for partitioning recommendations and troubleshooting failed Hive queries.

Using the native web interface for HiveServer2 provides access to Hive configuration settings, local logs, metrics, and information about sessions and queries. See HiveServer2 Web UI on page 45 for details about accessing, configuring, and using HiveServer2 Web UI.

For additional Hive documentation, see the Apache Hive wiki.

Upstream Information for Hive

Detailed Hive documentation is available on the Apache Software Foundation site on the Hive project page. For specific areas of the Apache Hive documentation, see:

- Hive Query Language (HiveQL) Manual (for SQL syntax)
- Apache Hive wiki
- User Documentation
- Administrator Documentation

Because Cloudera does not support all Hive features, for example ACID (transactions), always check external Hive documentation against the current version and supported features of Hive included in CDH distribution.

Hive has its own JIRA issue tracker.

Hive Installation

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade:

- Install the Cloudera `yum`, `zypper`/YaST or `apt` repository.
- Install or upgrade CDH 5 and make sure it is functioning correctly.

For instructions, see Installing the Latest CDH 5 Release and Upgrading Unmanaged CDH Using the Command Line.

Using Hive data in HBase is a common task. See Importing Data Into HBase.

For information about Hive on Spark, see Running Hive on Spark on page 43.

Use the following sections to install, update, and configure Hive.

About Hive

Apache Hive is a powerful data warehousing application for Hadoop. It enables you to access your data using HiveQL, a language similar to SQL.

Install Hive on your client machine(s) from which you submit jobs; you do not need to install it on the nodes in your Hadoop cluster. As of CDH 5, Hive supports HCatalog which must be installed separately.

HiveServer2

HiveServer2 is an improved version of HiveServer that supports a Thrift API tailored for JDBC and ODBC clients, Kerberos authentication, and multi-client concurrency. The CLI for HiveServer2 is Beeline.

Warning: Because of concurrency and security issues, HiveServer1 and the Hive CLI are deprecated in CDH 5 and will be removed in a future release. Cloudera recommends you migrate to Beeline and HiveServer2 as soon as possible. The Hive CLI is not needed if you are using Beeline with HiveServer2.

Upgrading Hive

Upgrade Hive on all the hosts on which it is running including both servers and clients.

Warning: Because of concurrency and security issues, HiveServer1 and the Hive CLI are deprecated in CDH 5 and will be removed in a future release. Cloudera recommends you migrate to Beeline and HiveServer2 as soon as possible. The Hive CLI is not needed if you are using Beeline with HiveServer2.

Note: To see which version of Hive is shipping in CDH 5, check the Version and Packaging Information. For important information on new and changed components, see the CDH 5 Release Notes.

Checklist to Help Ensure Smooth Upgrades

The following best practices for configuring and maintaining Hive will help ensure that upgrades go smoothly.

- Configure periodic backups of the metastore database. Use `mysqldump`, or the equivalent for your vendor if you are not using MySQL.

- Make sure `datanucleus.autoCreateSchema` is set to false (in all types of database) and `datanucleus.fixedDatastore` is set to true (for MySQL and Oracle) in *all* `hive-site.xml` files. See the configuration instructions for more information about setting the properties in `hive-site.xml`.
- Insulate the metastore database from users by running the metastore service in Remote mode. If you do not follow this recommendation, make sure you remove DROP, ALTER, and CREATE privileges from the Hive user configured in `hive-site.xml`. See Configuring the Hive Metastore on page 19 for complete instructions for each type of supported database.

> **Warning:**
>
> Make sure you have read and understood all incompatible changes and known issues before you upgrade Hive.

Upgrading Hive from a Lower Version of CDH 5

The instructions that follow assume that you are upgrading Hive as part of a CDH 5 upgrade, and have already performed the steps under Upgrading from an Earlier CDH 5 Release to the Latest Release.

> **Important:**
>
> - If you are currently running Hive under MRv1, check for the following property and value in `/etc/mapred/conf/mapred-site.xml`:
>
> ```
> <property>
>   <name>mapreduce.framework.name</name>
>   <value>yarn</value>
> </property>
> ```
>
>   Remove this property before you proceed; otherwise Hive queries spawned from MapReduce jobs will fail with a null pointer exception (NPE).
> - If you have installed the `hive-hcatalog-server` package in the past, you must remove it before you proceed; otherwise the upgrade will fail.
> - If you are upgrading Hive from CDH 5.0.5 to CDH 5.4, 5.3 or 5.2 on Debian 7.0, and a Sentry version higher than 5.0.4 and lower than 5.1.0 is installed, you must upgrade Sentry before upgrading Hive; otherwise the upgrade will fail. See Apache Hive Known Issues for more details.
> - CDH 5.2 and higher clients cannot communicate with CDH 5.1 and lower servers. This means that you must upgrade the server before the clients.

To upgrade Hive from a lower version of CDH 5, proceed as follows.

Step 1: Stop all Hive Processes and Daemons

> **Warning:**
>
> You **must** make sure no Hive processes are running. If Hive processes are running during the upgrade, the new version will not work correctly.

1. Stop any HiveServer processes that are running:

```
$ sudo service hive-server stop
```

2. Stop any HiveServer2 processes that are running:

```
$ sudo service hive-server2 stop
```

3. Stop the metastore:

```
$ sudo service hive-metastore stop
```

Step 2: Install the new Hive version on all hosts (Hive servers and clients)

SeeInstalling Hive on page 17

Step 3: Verify that the Hive Metastore is Properly Configured

See Configuring the Hive Metastore on page 19 for detailed instructions.

Step 4: Upgrade the Metastore Schema

Important:

- Cloudera recommends that you make a backup copy of your metastore database before running the `schematool` or the upgrade scripts. You might need this backup copy if there are problems during the upgrade or if you need to downgrade to a previous version.
- You *must* upgrade the metastore schema to the version corresponding to the new version of Hive before starting Hive after the upgrade. Failure to do so may result in metastore corruption.

To upgrade the Hive metastore schema, you can use either the Hive `schematool` or use the schema upgrade scripts that are provided with the Hive package. Cloudera recommends that you use the `schematool`.

Using Hive schematool (Recommended):

The Hive distribution includes a command-line tool for Hive metastore schema manipulation called `schematool`. This tool can be used to initialize the metastore schema for the current Hive version. It can also upgrade the schema from an older version to the current one. You must add properties to the `hive-site.xml` before you can use it. See Using the Hive Schema Tool on page 37 for information about how to set the tool up and for usage examples. To upgrade the schema, use the `upgradeSchemaFrom` option to specify the version of the schema you are currently using. For example, if you are upgrading a MySQL metastore schema from Hive 0.13.1, use the following syntax:

```
$ schematool -dbType mysql -passWord <db_user_pswd> -upgradeSchemaFrom
  0.13.1 -userName <db_user_name>
Metastore connection URL:
jdbc:mysql://<cluster_address>:3306/<user_name>?useUnicode=true&characterEncoding=UTF-8
Metastore Connection Driver : com.mysql.jdbc.Driver
Metastore connection User: <user_name>
Starting upgrade metastore schema from version 0.13.1 to <new_version>
Upgrade script upgrade-0.13.1-to-<new_version>.mysql.sql
Completed pre-0-upgrade-0.13.1-to-<new_version>.mysql.sql
Completed upgrade-0.13.1-to-<new_version>.mysql.sql
schemaTool completed
```

Note: The `upgradeSchemaFrom` option requires the Hive version and not the CDH version. See CDH 5 Packaging and Tarball Information for information about which Hive version ships with each CDH release.

Using Schema Upgrade Scripts:

Navigate to the directory where the schema upgrade scripts are located:

- If you installed CDH with parcels, the scripts are in the following location:

```
/opt/cloudera/parcels/CDH/lib/hive/scripts/metastore/upgrade/<database_name>
```

- If you installed CDH with packages, the scripts are in the following location:

```
/usr/lib/hive/scripts/metastore/upgrade/<database_name>
```

For example, if your Hive metastore is MySQL and you installed CDH with packages, navigate to `/usr/lib/hive/scripts/metastore/upgrade/mysql`.

Run the appropriate schema upgrade scripts in order. Start with the script for your database type and Hive version, and run all subsequent scripts.

For example, if you are currently running Hive 0.13.1 with MySQL and upgrading to Hive 1.1.0, start with the script for 0.13.0 to 0.14.0 for MySQL, and then run the script for Hive 0.14.0 to 1.1.0.

For more information about using the scripts to upgrade the schema, see the README in the directory with the scripts.

Step 5: Start the Metastore, HiveServer2, and Beeline

See:

- Starting the Metastore on page 34
- Starting, Stopping, and Using HiveServer2 on page 35

The upgrade is now complete.

Troubleshooting: if you failed to upgrade the metastore

If you failed to upgrade the metastore as instructed above, proceed as follows.

1. **Identify the problem.**

   The symptoms are as follows:

   - Hive stops accepting queries.
   - In a cluster managed by Cloudera Manager, the Hive Metastore canary fails.
   - An error such as the following appears in the Hive Metastore Server logs:

```
Hive Schema version 0.13.0 does not match metastore's schema version 0.12.0 Metastore
is not upgraded or corrupt.
```

2. **Resolve the problem.**

   If the problem you are having matches the symptoms just described, do the following:

   1. Stop all Hive services; for example:

```
$ sudo service hive-server2 stop
$ sudo service hive-metastore stop
```

   2. Run the Hive schematool, as instructed here.

      Make sure the value you use for the `-upgradeSchemaFrom` option matches the version you are *currently running* (not the new version). For example, if the error message in the log is

```
Hive Schema version 0.13.0 does not match metastore's schema version 0.12.0 Metastore
is not upgraded or corrupt.
```

then the value of `-upgradeSchemaFrom` must be `0.12.0`.

3. Restart the Hive services you stopped.

Installing Hive

Install the appropriate Hive packages using the appropriate command for your distribution.

OS	Command
RHEL-compatible	`$ sudo yum install <pkg1> <pkg2> ...`
SLES	`$ sudo zypper install <pkg1> <pkg2> ...`
Ubuntu or Debian	`$ sudo apt-get install <pkg1> <pkg2> ...`

The packages are:

- `hive` – base package that provides the complete language and runtime
- `hive-metastore` – provides scripts for running the metastore as a standalone service (optional)
- `hive-server2` – provides scripts for running HiveServer2
- `hive-hbase` - optional; install this package if you want to use Hive with HBase.

> **Important:** After installing Hive, see HiveServer2 Performance Best Practices on page 59 for information about optimizing your Hive deployment and your Hive workloads for best performance results.

Heap Size and Garbage Collection for Hive Components

This section provides guidelines for setting HiveServer2 and Hive metastore memory and garbage-collection properties.

Memory Recommendations

HiveServer2 and the Hive metastore require sufficient memory to run correctly. The default heap size of 256 MB for each component is inadequate for production workloads. Consider the following guidelines for sizing the heap for each component, based on your cluster size.

Number of Concurrent Connections	HiveServer2 Heap Size Recommended Range	Hive Metastore Heap Size Recommended Range
Up to 40 concurrent connections Cloudera recommends splitting HiveServer2 into multiple instances and load-balancing once you start allocating over 16 GB to HiveServer2. This reduces the impact of Java garbage collection on active processing by the service.	12 - 16 GB	12 - 16 GB
Up to 20 concurrent connections	6 - 12 GB	10 - 12 GB
Up to 10 concurrent connections	4 - 6 GB	4 - 10 GB
One connection	4 GB	4 GB

> **Important:** These numbers are general guidance only, and can be affected by factors such as number of columns, partitions, complex joins, and client activity. Based on your anticipated deployment, refine through testing to arrive at the best values for your environment.

In addition, the Beeline CLI should use a heap size of at least 2 GB.

Set the PermGen space for Java garbage collection to 512 MB for all.

Configuring Heap Size and Garbage Collection

Using Cloudera Manager

To configure heap size and garbage collection for HiveServer2:

1. To set heap size, go to **Home > Hive > Configuration > HiveServer2 > Resource Management**.
2. Set **Java Heap Size of HiveServer2 in Bytes** to the desired value, and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > HiveServer2 > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M`, the type of garbage collector used (`ConcMarkSweepGC` or `ParNewGC`), and enable or disable the garbage collection overhead limit in **Java Configuration Options for HiveServer2**.

   The following example sets the PermGen space to `512M`, uses the new Parallel Collector, and disables the garbage collection overhead limit:

```
-XX:MaxPermSize=512M -XX:+UseParNewGC -XX:-UseGCOverheadLimit
```

5. From the **Actions** drop-down menu, select `Restart` to restart the HiveServer2 service.

To configure heap size and garbage collection for the Hive metastore:

1. To set heap size, go to **Home > Hive > Configuration > Hive Metastore > Resource Management**.
2. Set **Java Heap Size of Hive Metastore Server in Bytes** to the desired value, and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > Hive Metastore Server > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M`, the type of garbage collector used (`ConcMarkSweepGC` or `ParNewGC`), and enable or disable the garbage collection overhead limit in **Java Configuration Options for Hive Metastore Server**. For an example of this setting, see step 4 above for configuring garbage collection for HiveServer2.
5. From the **Actions** drop-down menu, select `Restart` to restart the Hive Metastore service.

To configure heap size and garbage collection for the Beeline CLI:

1. To set heap size, go to **Home > Hive > Configuration > Gateway > Resource Management**.
2. Set **Client Java Heap Size in Bytes** to at least `2 GiB` and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > Gateway > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M` in **Client Java Configuration Options**.

   The following example sets the PermGen space to `512M` and specifies `IPv4`:

```
-XX:MaxPermSize=512M -Djava.net.preferIPv4Stack=true
```

5. From the **Actions** drop-down menu, select `Restart` to restart the client service.

Using the Command Line

To configure the heap size for HiveServer2 and Hive metastore, set the `-Xmx` parameter in the `HADOOP_OPTS` variable to the desired maximum heap size in `/etc/hive/hive-env.sh`.

To configure the heap size for the Beeline CLI, set the `HADOOP_HEAPSIZE` environment variable in `/etc/hive/hive-env.sh` before starting the Beeline CLI.

The following example shows a configuration with the following settings:

- HiveServer2 uses 12 GB heap.
- Hive metastore uses 12 GB heap.
- Hive clients use 2 GB heap.

The settings to change are in bold. All of these lines are commented out (prefixed with a # character) by default.

```
if [ "$SERVICE" = "cli" ]; then
  if [ -z "$DEBUG" ]; then
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms12288m
               -XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:+UseParNewGC
-XX:-UseGCOverheadLimit"
  else
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms12288m
               -XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:-UseGCOverheadLimit"
  fi
fi

export HADOOP_HEAPSIZE=2048
```

You can use either the Concurrent Collector or the new Parallel Collector for garbage collection by passing `-XX:+useConcMarkSweepGC` or `-XX:+useParNewGC` in the `HADOOP_OPTS` lines above. To enable the garbage collection overhead limit, remove the `-XX:-UseGCOverheadLimit` setting or change it to `-XX:+UseGCOverheadLimit`.

Set the PermGen space for Java garbage collection to `512M` for all in the `JAVA-OPTS` environment variable. For example:

```
set JAVA_OPTS="-Xms256m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m"
```

Configuration for WebHCat

If you want to use WebHCat, you need to set the `PYTHON_CMD` variable in `/etc/default/hive-webhcat-server` after installing Hive; for example:

```
export PYTHON_CMD=/usr/bin/python
```

Configuring the Hive Metastore

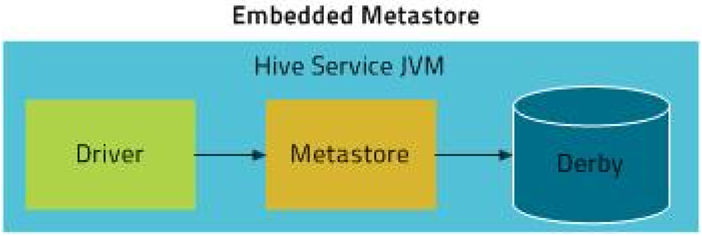
The Hive metastore service stores the metadata for Hive tables and partitions in a relational database, and provides clients (including Hive) access to this information using the metastore service API. This page explains deployment options and provides instructions for setting up a database in a recommended configuration.

Metastore Deployment Modes

Note: On this page, **HiveServer**, refers to HiveServer1 or HiveServer2, whichever you are using.

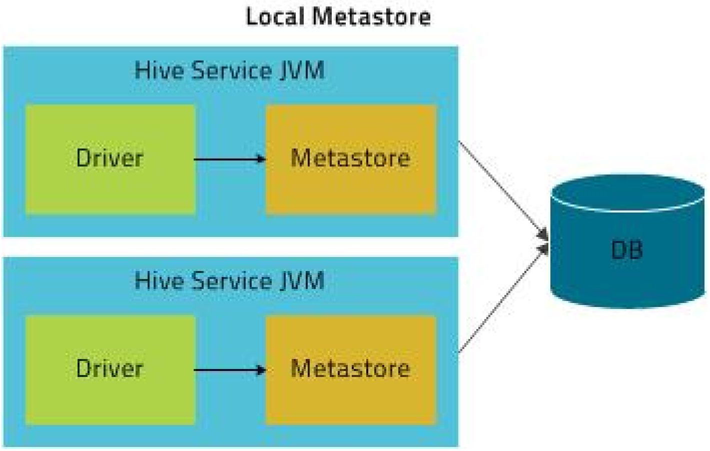
Embedded Mode

Cloudera recommends using this mode for experimental purposes only.



Embedded mode is the default metastore deployment mode for CDH. In this mode, the metastore uses a Derby database, and both the database and the metastore service are embedded in the main HiveServer process. Both are started for you when you start the HiveServer process. This mode requires the least amount of effort to configure, but it can support only one active user at a time and is not certified for production use.

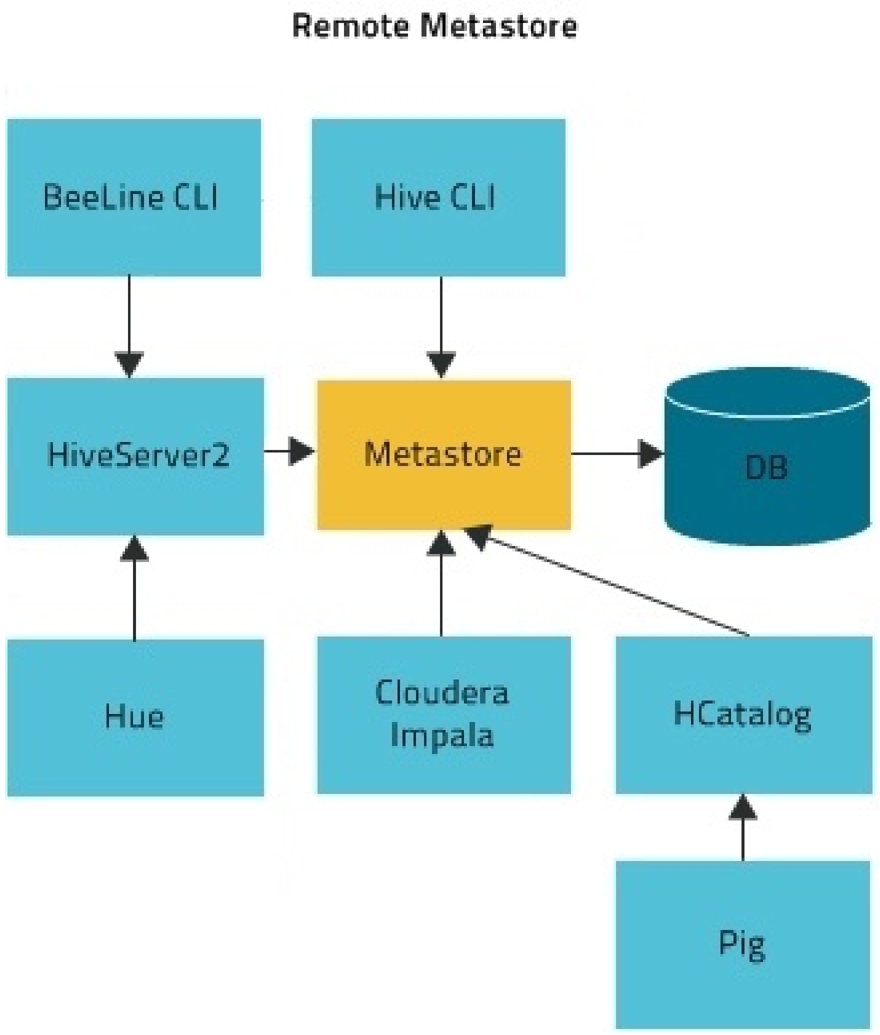
Local Mode



In Local mode, the Hive metastore service runs in the same process as the main HiveServer process, but the metastore database runs in a separate process, and can be on a separate host. The embedded metastore service communicates with the metastore database over JDBC.

Remote Mode

Cloudera recommends that you use this mode.



In Remote mode, the Hive metastore service runs in its own JVM process. HiveServer2, HCatalog, Impala, and other processes communicate with it using the Thrift network API (configured using the `hive.metastore.uris` property). The metastore service communicates with the metastore database over JDBC (configured using the `javax.jdo.option.ConnectionURL` property). The database, the HiveServer process, and the metastore service can all be on the same host, but running the HiveServer process on a separate host provides better availability and scalability.

The main advantage of Remote mode over Local mode is that Remote mode does not require the administrator to share JDBC login information for the metastore database with each Hive user. HCatalog requires this mode.

Supported Metastore Databases

For up-to-date information, see CDH and Cloudera Manager Supported Databases. Cloudera strongly encourages you to use MySQL because it is the most popular with the rest of the Hive user community, and, hence, receives more testing than the other options. For installation information, see:

- MySQL Database
- External PostgreSQL Database
- Oracle Database

In addition to the above supported databases, CDH also supports using MariaDB. For information about installing and configuring MariaDB for CDH, see MariaDB Database.

Metastore Memory Requirements

Number of Concurrent Connections	HiveServer2 Heap Size Recommended Range	Hive Metastore Heap Size Recommended Range
Up to 40 concurrent connections Cloudera recommends splitting HiveServer2 into multiple instances and load-balancing once you start allocating over 16 GB to HiveServer2. This reduces the impact of Java garbage collection on active processing by the service.	12 - 16 GB	12 - 16 GB
Up to 20 concurrent connections	6 - 12 GB	10 - 12 GB
Up to 10 concurrent connections	4 - 6 GB	4 - 10 GB
One connection	4 GB	4 GB

Important: These numbers are general guidance only, and can be affected by factors such as number of columns, partitions, complex joins, and client activity. Based on your anticipated deployment, refine through testing to arrive at the best values for your environment.

For information on configuring heap for Hive MetaStore, as well as HiveServer2 and Hive clients, see Heap Size and Garbage Collection for Hive Components on page 17.

Configuring the Metastore Database

This section describes how to configure Hive to use a remote database, with examples for MySQL, PostgreSQL, and Oracle.

The configuration properties for the Hive metastore are documented in the Hive Metastore Administration documentation on the Apache wiki.

Note: For information about additional configuration that may be needed in a secure cluster, see Hive Authentication on page 86.

Configuring a Remote MySQL Database for the Hive Metastore

Cloudera recommends you configure a database for the metastore on one or more remote servers (that is, on a host or hosts separate from the HiveServer1 or HiveServer2 process). MySQL is the most popular database to use. Proceed as follows.

1. **Install and start MySQL if you have not already done so**

   To install MySQL on a RHEL system:

```
$ sudo yum install mysql-server
```

   To install MySQL on a SLES system:

```
$ sudo zypper install mysql
$ sudo zypper install libmysqlclient_r17
```

   To install MySQL on a Debian/Ubuntu system:

```
$ sudo apt-get install mysql-server
```

After using the command to install MySQL, you may need to respond to prompts to confirm that you do want to complete the installation. After installation completes, start the `mysql` daemon.

On RHEL systems

```
$ sudo service mysqld start
```

On SLES and Debian/Ubuntu systems

```
$ sudo service mysql start
```

2. **Configure the MySQL service and connector**

Before you can run the Hive metastore with a remote MySQL database, you must configure a connector to the remote MySQL database, set up the initial database schema, and configure the MySQL user account for the Hive user.

To install the MySQL connector on a RHEL 6 system:

On the Hive Metastore server host, install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo yum install mysql-connector-java
$ ln -s /usr/share/java/mysql-connector-java.jar
/usr/lib/hive/lib/mysql-connector-java.jar
```

To install the MySQL connector on a RHEL 5 system:

Download the MySQL JDBC driver from http://www.mysql.com/downloads/connector/j/5.1.html. You will need to sign up for an account if you do not already have one, and log in, before you can download it. Then copy it to the `/usr/lib/hive/lib/` directory. For example:

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar
/usr/lib/hive/lib/
```

> **Note:** At the time of publication, *version* was `5.1.31`, but the version may have changed by the time you read this. If you are using MySQL version 5.6, you must use version 5.1.26 or higher of the driver.

To install the MySQL connector on a SLES system:

On the Hive Metastore server host, install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo zypper install mysql-connector-java
$ ln -s /usr/share/java/mysql-connector-java.jar
/usr/lib/hive/lib/mysql-connector-java.jar
```

To install the MySQL connector on a Debian/Ubuntu system:

On the Hive Metastore server host, install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo apt-get install libmysql-java
$ ln -s /usr/share/java/libmysql-java.jar /usr/lib/hive/lib/libmysql-java.jar
```

Configure MySQL to use a strong password and to start at boot. Note that in the following procedure, your current `root` password is blank. Press the Enter key when you're prompted for the root password.

To set the MySQL root password:

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

To make sure the MySQL server starts at boot:

- On RHEL systems:

```
$ sudo /sbin/chkconfig mysqld on
$ sudo /sbin/chkconfig --list mysqld
mysqld          0:off   1:off   2:on    3:on    4:on    5:on    6:off
```

- On SLES systems:

```
$ sudo chkconfig --add mysql
```

- On Debian/Ubuntu systems:

```
$ sudo chkconfig mysql on
```

3. Create the database and user

The instructions in this section assume you are using Remote mode, and that the MySQL database is installed on a separate host from the metastore service, which is running on a host named `metastorehost` in the example.

> **Note:**
>
> If the metastore service will run on the host where the database is installed, replace `'metastorehost'` in the `CREATE USER` example with `'localhost'`. Similarly, the value of `javax.jdo.option.ConnectionURL` in `/etc/hive/conf/hive-site.xml` (discussed in the next step) must be `jdbc:mysql://localhost/metastore`. For more information on adding MySQL users, see http://dev.mysql.com/doc/refman/5.5/en/adding-users.html.

Create the initial database schema. Cloudera recommends using the Hive schema tool to do this.

If for some reason you decide not to use the schema tool, you can use the `hive-schema-`*`n.n.n`*`.mysql.sql` file instead; that file is located in the `/usr/lib/hive/scripts/metastore/upgrade/mysql/` directory. (*n.n.n* is the current Hive version, for example 1.1.0.) Proceed as follows if you decide to use `hive-schema-`*`n.n.n`*`.mysql.sql`.

Example using hive-schema-*n.n.n*mysql.sql

Note:

Do this only if you are not using the Hive schema tool.

```
$ mysql -u root -p
Enter password:
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE /usr/lib/hive/scripts/metastore/upgrade/mysql/hive-schema-n.n.n.mysql.sql;
```

You also need a MySQL user account for Hive to use to access the metastore. It is very important to prevent this user account from creating or altering tables in the metastore database schema.

Important: To prevent users from inadvertently corrupting the metastore schema when they use lower or higher versions of Hive, set the `hive.metastore.schema.verification` property to true in `/usr/lib/hive/conf/hive-site.xml` on the metastore host.

Example

```
mysql> CREATE USER 'hive'@'metastorehost' IDENTIFIED BY 'mypassword';
...
mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'hive'@'metastorehost';
mysql> GRANT ALL PRIVILEGES ON metastore.* TO 'hive'@'metastorehost';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

4. **Configure the metastore service to communicate with the MySQL database**

This step shows the configuration properties you need to set in `hive-site.xml` (`/usr/lib/hive/conf/hive-site.xml`) to configure the metastore service to communicate with the MySQL database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

Given a MySQL database running on `myhost` and the user account `hive` with the password `mypassword`, set the configuration as follows (overwriting any existing values).

Note:

The `hive.metastore.local` property is no longer supported (as of Hive 0.10); setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://myhost/metastore</value>
  <description>the URL of the MySQL database</description>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
```

```
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>

<property>
  <name>datanucleus.autoStartMechanism</name>
  <value>SchemaTable</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>

<property>
<name>hive.metastore.schema.verification</name>
<value>true</value>
</property>
```

Configuring a Remote PostgreSQL Database for the Hive Metastore

Before you can run the Hive metastore with a remote PostgreSQL database, you must configure a connector to the remote PostgreSQL database, set up the initial database schema, and configure the PostgreSQL user account for the Hive user.

1. **Install and start PostgreSQL if you have not already done so**

   To install PostgreSQL on a RHEL system:

```
$ sudo yum install postgresql-server
```

   To install PostgreSQL on a SLES system:

```
$ sudo zypper install postgresql-server
```

   To install PostgreSQL on a Debian/Ubuntu system:

```
$ sudo apt-get install postgresql
```

   After using the command to install PostgreSQL, you may need to respond to prompts to confirm that you do want to complete the installation. In order to finish installation on RHEL compatible systems, you need to initialize the database. Please note that this operation is not needed on Ubuntu and SLES systems as it's done automatically on first start:

   To initialize database files on RHEL compatible systems

```
$ sudo service postgresql initdb
```

   To ensure that your PostgreSQL server will be accessible over the network, you need to do some additional configuration.

   First you need to edit the `postgresql.conf` file. Set the `listen_addresses` property to `*`, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the `standard_conforming_strings` property is set to `off`.

You can check that you have the correct values as follows:

On Red-Hat-compatible systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf  | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

On SLES systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf  | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

On Ubuntu and Debian systems:

```
$ cat /etc/postgresql/9.1/main/postgresql.conf | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

You also need to configure authentication for your network in `pg_hba.conf`. You need to make sure that the PostgreSQL user that you will create later in this procedure will have access to the server from a remote host. To do this, add a new line into `pg_hba.con` that has the following information:

```
host    <database>         <user>          <network address>         <mask>
   md5
```

The following example allows all users to connect from all hosts to all your databases:

```
host    all         all         0.0.0.0         0.0.0.0               md5
```

Note:

This configuration is applicable only for a network listener. Using this configuration does not open all your databases to the entire world; the user must still supply a password to authenticate himself, and privilege restrictions configured in PostgreSQL will still be applied.

After completing the installation and configuration, you can start the database server:

Start PostgreSQL Server

```
$ sudo service postgresql start
```

Use `chkconfig` utility to ensure that your PostgreSQL server will start at a boot time. For example:

```
chkconfig postgresql on
```

You can use the `chkconfig` utility to verify that PostgreSQL server will be started at boot time, for example:

```
chkconfig --list postgresql
```

2. **Install the PostgreSQL JDBC driver**

   Before you can run the Hive metastore with a remote PostgreSQL database, you must configure a JDBC driver to the remote PostgreSQL database, set up the initial database schema, and configure the PostgreSQL user account for the Hive user.

   To install the PostgreSQL JDBC Driver on a RHEL 6 system:

On the Hive Metastore server host, install `postgresql-jdbc` package and create symbolic link to the `/usr/lib/hive/lib/` directory. For example:

```
$ sudo yum install postgresql-jdbc
$ ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-jdbc.jar
```

To install the PostgreSQL connector on a RHEL 5 system:

You need to manually download the PostgreSQL connector from http://jdbc.postgresql.org/download.html and move it to the `/usr/lib/hive/lib/` directory. For example:

```
$ wget http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar
$ mv postgresql-9.2-1002.jdbc4.jar /usr/lib/hive/lib/
```

> **Note:**
>
> You may need to use a different version if you have a different version of Postgres. You can check the version as follows:
>
> ```
> $ sudo rpm -qa | grep postgres
> ```

To install the PostgreSQL JDBC Driver on a SLES system:

On the Hive Metastore server host, install `postgresql-jdbc` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo zypper install postgresql-jdbc
$ ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-jdbc.jar
```

To install the PostgreSQL JDBC Driver on a Debian/Ubuntu system:

On the Hive Metastore server host, install `libpostgresql-jdbc-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo apt-get install libpostgresql-jdbc-java
$ ln -s /usr/share/java/postgresql-jdbc4.jar /usr/lib/hive/lib/postgresql-jdbc4.jar
```

3. **Create the metastore database and user account**

Proceed as in the following example, using the appropriate script in `/usr/lib/hive/scripts/metastore/upgrade/postgres/` *n.n.n* is the current Hive version, for example 1.1.0:

```
$ sudo -u postgres psql
postgres=# CREATE USER hiveuser WITH PASSWORD 'mypassword';
postgres=# CREATE DATABASE metastore;
postgres=# \c metastore;
You are now connected to database 'metastore'.
postgres=# \i
/usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schema-n.n.n.postgres.sql
SET
SET
...
```

Now you need to grant permission for all metastore tables to user `hiveuser`. PostgreSQL does not have statements to grant the permissions for all tables at once; you'll need to grant the permissions one table at a time. You could automate the task with the following SQL script:

Note:

If you are running these commands interactively and are still in the Postgres session initiated at the beginning of this step, you do not need to repeat `sudo -u postgres psql`.

```
bash# sudo -u postgres psql
metastore=# \c metastore
metastore=# \pset tuples_only on
metastore=# \o /tmp/grant-privs
metastore=#   SELECT 'GRANT SELECT,INSERT,UPDATE,DELETE ON "'  || schemaname || '". "'
 ||tablename ||'" TO hiveuser ;'
metastore-#   FROM pg_tables
metastore-#   WHERE tableowner = CURRENT_USER and schemaname = 'public';
metastore=# \o
metastore=# \pset tuples_only off
metastore=# \i /tmp/grant-privs
```

You can verify the connection from the machine where you'll be running the metastore service as follows:

```
psql -h myhost -U hiveuser -d metastore
metastore=#
```

4. **Configure the metastore service to communicate with the PostgreSQL database**

   This step shows the configuration properties you need to set in `hive-site.xml` (`/usr/lib/hive/conf/hive-site.xml`) to configure the metastore service to communicate with the PostgreSQL database. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

   Given a PostgreSQL database running on host `myhost` under the user account `hive` with the password `mypassword`, you would set configuration properties as follows.

Note:

- The instructions in this section assume you are using Remote mode, and that the PostgreSQL database is installed on a separate host from the metastore server.
- The `hive.metastore.local` property is no longer supported as of Hive 0.10; setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://myhost/metastore</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hiveuser</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>

<property>
<name>hive.metastore.schema.verification</name>
<value>true</value>
</property>
```

5. **Test connectivity to the metastore**

```
$ hive –e “show tables;”
```

> **Note:** This will take a while the first time.

Configuring a Remote Oracle Database for the Hive Metastore

Before you can run the Hive metastore with a remote Oracle database, you must configure a connector to the remote Oracle database, set up the initial database schema, and configure the Oracle user account for the Hive user.

1. **Install and start Oracle**

   The Oracle database is not part of any Linux distribution and must be purchased, downloaded and installed separately. You can use the Express edition, which can be downloaded free from the Oracle website.

2. **Install the Oracle JDBC Driver**

   You must download the Oracle JDBC Driver from the Oracle website and put the JDBC JAR file into the `/usr/lib/hive/lib/` directory. For example, the version 6 JAR file is named `ojdbc6.jar`. To download the JDBC driver, visit the Oracle JDBC and UCP Downloads page, and click on the link for your Oracle Database version. Download the `ojdbc6.jar` file.

   > **Note:** This URLs was correct at the time of publication, but can change.

```
sudo mv ojdbc<version_number>.jar /usr/lib/hive/lib/
```

3. **Create the Metastore database and user account**

   Connect to your Oracle database as an administrator and create the user that will use the Hive metastore.

```
$ sqlplus "sys as sysdba"
SQL> create user hiveuser identified by mypassword;
SQL> grant connect to hiveuser;
SQL> grant all privileges to hiveuser;
```

   Connect as the newly created `hiveuser` user and load the initial schema, as in the following example. Use the appropriate script for the current release (for example hive-schema-1.1.0.oracle.sql) in `/usr/lib/hive/scripts/metastore/upgrade/oracle/` :

```
$ sqlplus hiveuser
SQL> @/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-n.n.n.oracle.sql
```

Connect back as an administrator and remove the power privileges from user `hiveuser`. Then grant limited access to all the tables:

```
$ sqlplus "sys as sysdba"
SQL> revoke all privileges from hiveuser;
SQL> BEGIN
  2     FOR R IN (SELECT owner, table_name FROM all_tables WHERE owner='HIVEUSER') LOOP

  3        EXECUTE IMMEDIATE 'grant  SELECT,INSERT,UPDATE,DELETE on
'||R.owner||'.'||R.table_name||' to hiveuser';
  4     END LOOP;
  5  END;
  6
  7  /
```

4. Configure the Metastore Service to Communicate with the Oracle Database

This step shows the configuration properties you need to set in `hive-site.xml` (`/usr/lib/hive/conf/hive-site.xml`) to configure the metastore service to communicate with the Oracle database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that must be configured on all of them; the others are used only on the metastore host.

Example

Given an Oracle database running on `myhost` and the user account `hiveuser` with the password `mypassword`, set the configuration as follows (overwriting any existing values):

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:oracle:thin:@//myhost/xe</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>oracle.jdbc.OracleDriver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hiveuser</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>

<property>
<name>hive.metastore.schema.verification</name>
<value>true</value>
</property>
```

Configuring HiveServer2

You must make the following configuration changes before using HiveServer2. Failure to do so may result in unpredictable behavior.

Warning: HiveServer1 is deprecated in CDH 5.3, and will be removed in a future release of CDH. Users of HiveServer1 should upgrade to HiveServer2 as soon as possible.

HiveServer2 Memory Requirements

Number of Concurrent Connections	HiveServer2 Heap Size Recommended Range	Hive Metastore Heap Size Recommended Range
Up to 40 concurrent connections Cloudera recommends splitting HiveServer2 into multiple instances and load-balancing once you start allocating over 16 GB to HiveServer2. This reduces the impact of Java garbage collection on active processing by the service.	12 - 16 GB	12 - 16 GB
Up to 20 concurrent connections	6 - 12 GB	10 - 12 GB
Up to 10 concurrent connections	4 - 6 GB	4 - 10 GB
One connection	4 GB	4 GB

Important: These numbers are general guidance only, and can be affected by factors such as number of columns, partitions, complex joins, and client activity. Based on your anticipated deployment, refine through testing to arrive at the best values for your environment.

For information on configuring heap for HiveServer2, as well as Hive metastore and Hive clients, see Heap Size and Garbage Collection for Hive Components on page 17 and the following video:

After you start the video, click **YouTube** in the lower right corner of the player window to watch it on YouTube where you can resize it for clearer viewing.

Figure 1: Troubleshooting HiveServer2 Service Crashes

Table Lock Manager (Required)

You must properly configure and enable Hive's Table Lock Manager. This requires installing ZooKeeper and setting up a ZooKeeper ensemble; see ZooKeeper Installation.

Important: Failure to do this will prevent HiveServer2 from handling concurrent query requests and may result in data corruption.

Enable the lock manager by setting properties in `/etc/hive/conf/hive-site.xml` as follows (substitute your actual ZooKeeper node names for those in the example):

```
<property>
  <name>hive.support.concurrency</name>
  <description>Enable Hive's Table Lock Manager Service</description>
  <value>true</value>
</property>
```

```
<property>
  <name>hive.zookeeper.quorum</name>
  <description>Zookeeper quorum used by Hive's Table Lock Manager</description>
  <value>zk1.myco.com,zk2.myco.com,zk3.myco.com</value>
</property>
```

> **Important:** Enabling the Table Lock Manager without specifying a list of valid Zookeeper quorum nodes will result in unpredictable behavior. Make sure that both properties are properly configured.

(The above settings are also needed if you are still using HiveServer1. HiveServer1 is deprecated; migrate to HiveServer2 as soon as possible.)

`hive.zookeeper.client.port`

If ZooKeeper is not using the default value for `ClientPort`, you need to set `hive.zookeeper.client.port` in `/etc/hive/conf/hive-site.xml` to the same value that ZooKeeper is using. Check `/etc/zookeeper/conf/zoo.cfg` to find the value for `ClientPort`. If `ClientPort` is set to any value other than 2181 (the default), set`hive.zookeeper.client.port` to the same value. For example, if `ClientPort` is set to 2222, set `hive.zookeeper.client.port` to 2222 as well:

```
<property>
  <name>hive.zookeeper.client.port</name>
  <value>2222</value>
  <description>
  The port at which the clients will connect.
  </description>
</property>
```

JDBC driver

The connection URL format and the driver class are different for HiveServer2 and HiveServer1:

HiveServer version	Connection URL	Driver Class
HiveServer2	`jdbc:hive2://<host>:<port>`	`org.apache.hive.jdbc.HiveDriver`
HiveServer1	`jdbc:hive://<host>:<port>`	`org.apache.hadoop.hive.jdbc.HiveDriver`

Authentication

HiveServer2 can be configured to authenticate all connections; by default, it allows any client to connect. HiveServer2 supports either Kerberos or LDAP authentication; configure this in the `hive.server2.authentication` property in the `hive-site.xml` file. You can also configure Pluggable Authentication on page 90, which allows you to use a custom authentication provider for HiveServer2; and HiveServer2 Impersonation on page 90, which allows users to execute queries and access HDFS files as the connected user rather than the super user who started the HiveServer2 daemon. For more information, see Hive Security Configuration.

Running HiveServer2 and HiveServer Concurrently

> **Warning:** Because of concurrency and security issues, HiveServer1 and the Hive CLI are deprecated in CDH 5 and will be removed in a future release. Cloudera recommends you migrate to Beeline and HiveServer2 as soon as possible. The Hive CLI is not needed if you are using Beeline with HiveServer2.

HiveServer2 and HiveServer1 can be run concurrently on the same system, sharing the same data sets. This allows you to run HiveServer1 to support, for example, Perl or Python scripts that use the native HiveServer1 Thrift bindings.

Both HiveServer2 and HiveServer1 bind to port 10000 by default, so at least one of them must be configured to use a different port. You can set the port for HiveServer2 in `hive-site.xml` by means of the `hive.server2.thrift.port` property. For example:

```
<property>
  <name>hive.server2.thrift.port</name>
  <value>10001</value>
  <description>TCP port number to listen on, default 10000</description>
</property>
```

You can also specify the port (and the host IP address in the case of HiveServer2) by setting these environment variables:

HiveServer version	Port	Host Address
HiveServer2	HIVE_SERVER2_THRIFT_PORT	HIVE_SERVER2_THRIFT_BIND_HOST
HiveServer1	HIVE_PORT	< *Host bindings cannot be specified* >

Starting the Metastore

Cloudera recommends that you deploy the Hive metastore, which stores the metadata for Hive tables and partitions, in "remote mode." In this mode the metastore service runs in its own JVM process and other services, such as HiveServer2, HCatalog, and Apache Impala communicate with the metastore using the Thrift network API.

Important:

If you are running the metastore in Remote mode, you **must** start the metastore before starting HiveServer2.

After installing and configuring the Hive metastore, you can start the service.

To run the metastore as a daemon, the command is:

```
$ sudo service hive-metastore start
```

File System Permissions

Your Hive data is stored in HDFS, normally under `/user/hive/warehouse`. The `/user/hive` and `/user/hive/warehouse` directories need to be created if they do not already exist. Make sure this location (or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`) exists and is writable by the users whom you expect to be creating tables.

Important: If you are using Sentry, do not follow the instructions on this page. See Before Enabling the Sentry Service for information on how to set up the Hive warehouse directory permissions for use with Sentry.

In addition, each user submitting queries must have an HDFS home directory. `/tmp` (on the local file system) must be world-writable, as Hive makes extensive use of it.

HiveServer2 Impersonation on page 90 allows users to execute queries and access HDFS files as the connected user.

If you do not enable impersonation, HiveServer2 by default executes all Hive tasks as the user ID that starts the Hive server; for clusters that use Kerberos authentication, this is the ID that maps to the Kerberos principal used with HiveServer2. Setting permissions to `1777`, as recommended above, allows this user access to the Hive warehouse directory.

You can change this default behavior by setting `hive.metastore.execute.setugi` to `true` *on both the server and client.* This setting causes the metastore server to use the client's user and group permissions.

Starting, Stopping, and Using HiveServer2

HiveServer2 is an improved version of HiveServer that supports Kerberos authentication and multi-client concurrency.

Warning:

If you are running the metastore in Remote mode, you must start the Hive metastore before you start HiveServer2. HiveServer2 tries to communicate with the metastore as part of its initialization bootstrap. If it is unable to do this, it fails with an error.

To start HiveServer2:

```
$ sudo service hive-server2 start
```

To stop HiveServer2:

```
$ sudo service hive-server2 stop
```

To confirm that HiveServer2 is working, start the `beeline` CLI and use it to execute a `SHOW TABLES` query on the HiveServer2 process:

```
$ /usr/lib/hive/bin/beeline
beeline> !connect jdbc:hive2://localhost:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000> SHOW TABLES;
show tables;
+-----------+
| tab_name  |
+-----------+
+-----------+
No rows selected (0.238 seconds)
0: jdbc:hive2://localhost:10000>
```

Using the Beeline CLI

Beeline is the CLI (command-line interface) developed specifically to interact with HiveServer2. It is based on the SQLLine CLI written by Marc Prud'hommeaux.

Note:

Cloudera does not currently support using the Thrift HTTP protocol to connect Beeline to HiveServer2 (meaning that you cannot set `hive.server2.transport.mode=http`). Use the Thrift TCP protocol.

Use the following commands to start `beeline` and connect to a running HiveServer2 process. In this example the HiveServer2 process is running on `localhost` at port `10000`:

```
$ beeline
beeline> !connect jdbc:hive2://localhost:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000>
```

Note:

If you are using HiveServer2 on a cluster that does *not* have Kerberos security enabled, then the password is arbitrary in the command for starting Beeline.

If you are using HiveServer2 on a cluster that does have Kerberos security enabled, see HiveServer2 Security Configuration on page 86.

As of CDH 5.2, there are still some Hive CLI features that are *not* available with Beeline. For example:

- Beeline does not show query logs like the Hive CLI
- When adding JARs to HiveServer2 with Beeline, the JARs must be on the HiveServer2 host.

At present the best source for documentation on Beeline is the original SQLLine documentation.

Starting HiveServer1 and the Hive Console

Warning: Because of concurrency and security issues, HiveServer1 and the Hive CLI are deprecated in CDH 5 and will be removed in a future release. Cloudera recommends you migrate to Beeline and HiveServer2 as soon as possible. The Hive CLI is not needed if you are using Beeline with HiveServer2.

To start HiveServer1:

```
$ sudo service hiveserver start
```

See also Running HiveServer2 and HiveServer Concurrently on page 33.

To start the Hive console:

```
$ hive
hive>
```

To confirm that Hive is working, issue the `show tables;` command to list the Hive tables; be sure to use a semi-colon after the command:

```
hive> show tables;
OK
Time taken: 10.345 seconds
```

Using Hive with HBase

To allow Hive scripts to use HBase, proceed as follows.

1. Install the `hive-hbase` package.
2. Add the following statements to the top of each script. Replace the `<Guava_version>` string with the current version numbers for Guava. (You can find current version numbers for CDH dependencies such as Guava in CDH's root `pom.xml` file for the current release, for example cdh-root-5.0.0.pom.)

```
ADD JAR /usr/lib/hive/lib/zookeeper.jar;
ADD JAR /usr/lib/hive/lib/hive-hbase-handler.jar
ADD JAR /usr/lib/hive/lib/guava-<Guava_version>.jar;
ADD JAR /usr/lib/hive/lib/hbase-client.jar;
ADD JAR /usr/lib/hive/lib/hbase-common.jar;
ADD JAR /usr/lib/hive/lib/hbase-hadoop-compat.jar;
ADD JAR /usr/lib/hive/lib/hbase-hadoop2-compat.jar;
ADD JAR /usr/lib/hive/lib/hbase-protocol.jar;
ADD JAR /usr/lib/hive/lib/hbase-server.jar;
ADD JAR /usr/lib/hive/lib/htrace-core.jar;
```

Using the Hive Schema Tool

Schema Version Verification

Hive now records the schema version in the metastore database and verifies that the metastore schema version is compatible with the Hive binaries that are going to access the metastore. The Hive properties to implicitly create or alter the existing schema are disabled by default. Hence, Hive will not attempt to change the metastore schema implicitly. When you execute a Hive query against an old schema, it will fail to access the metastore displaying an error message as follows:

```
$ build/dist/bin/hive -e "show tables"
FAILED: Execution Error, return code 1 from org.apache.hadoop.hive.ql.exec.DDLTask.
java.lang.RuntimeException: Unable to instantiate
org.apache.hadoop.hive.metastore.HiveMetaStoreClient
```

The error log will contain an entry similar to the following:

```
...
Caused by: MetaException(message:Version information not found in metastore. )
       at org.apache.hadoop.hive.metastore.ObjectStore.checkSchema(ObjectStore.java:5638)
...
```

To suppress the schema check and allow the metastore to implicitly modify the schema, you need to set the `hive.metastore.schema.verification` configuration property to `false` in `hive-site.xml`.

Using schematool

Use the Hive `schematool` to initialize the metastore schema for the current Hive version or to upgrade the schema from an older version. The tool tries to find the current schema from the metastore if it is available there.

The `schematool` determines the SQL scripts that are required to initialize or upgrade the schema and then executes those scripts against the backend database. The metastore database connection information such as JDBC URL, JDBC driver and database credentials are extracted from the Hive configuration. You can provide alternate database credentials if needed.

The following options are available as part of the `schematool` package.

```
$ schematool -help
usage: schemaTool
 -dbType <databaseType>             Metastore database type
 -dryRun                            List SQL scripts (no execute)

 -help                              Print this message
 -info                              Show config and schema details
 -initSchema                        Schema initialization
 -initSchemaTo <initTo>             Schema initialization to a version
 -passWord <password>               Override config file password
 -upgradeSchema                     Schema upgrade
 -upgradeSchemaFrom <upgradeFrom>   Schema upgrade from a version
 -userName <user>                   Override config file user name
 -verbose                           Only print SQL statements
```

The `dbType` option should always be specified and can be one of the following:

```
 derby|mysql|postgres|oracle
```

Usage Examples

- Initialize your metastore to the current schema for a new Hive setup using the `initSchema` option.

```
$ schematool -dbType derby -initSchema
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting metastore schema initialization to <new_version>
Initialization script hive-schema-<new_version>.derby.sql
Initialization script completed
schemaTool completed
```

- Get schema information using the `info` option.

```
$ schematool -dbType derby -info
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Hive distribution version:       <new_version>
Required schema version:         <new_version>
Metastore schema version:        <new_version>
schemaTool completed
```

- If you attempt to get schema information from older metastores that did not store version information, the tool will report an error as follows.

```
$ schematool -dbType derby -info
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Hive distribution version:       <new_version>
Required schema version:         <new_version>
org.apache.hadoop.hive.metastore.HiveMetaException: Failed to get schema version.
*** schemaTool failed ***
```

- You can upgrade schema from a CDH 4 release by specifying the `upgradeSchemaFrom` option.

```
$ schematool -dbType derby -upgradeSchemaFrom 0.10.0
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting upgrade metastore schema from version 0.10.0 to <new_version>
Upgrade script upgrade-0.10.0-to-<new_version>.derby.sql
Completed upgrade-0.10.0-to-<new_version>.derby.sql
Upgrade script upgrade-0.11.0-to-<new_version>.derby.sql
Completed upgrade-0.11.0-to-<new_version>.derby.sql
schemaTool completed
```

The Hive versions of the older CDH releases are:

CDH Releases	Hive Version
CDH 3	0.7.0
CDH 4.0	0.8.0
CDH 4.1	0.9.0
CDH 4.2 and higher	0.10.0

- If you want to find out all the required scripts for a schema upgrade, use the `dryRun` option.

```
$ build/dist/bin/schematool -dbType derby -upgradeSchemaFrom 0.7.0 -dryRun
13/09/27 17:06:31 WARN conf.Configuration: hive.server2.enable.impersonation is
deprecated. Instead, use hive.server2.enable.doAs
Metastore connection URL:        jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
Starting upgrade metastore schema from version 0.7.0 to <new_version>
Upgrade script upgrade-0.7.0-to-0.8.0.derby.sql
Upgrade script upgrade-0.8.0-to-0.9.0.derby.sql
Upgrade script upgrade-0.9.0-to-0.10.0.derby.sql
Upgrade script upgrade-0.10.0-to-0.11.0.derby.sql
Upgrade script upgrade-0.11.0-to-<new_version>.derby.sql
schemaTool completed
```

Installing the Hive JDBC Driver on Clients

To access the Hive server with JDBC clients, such as Beeline, install the JDBC driver for HiveServer2 that is defined in `org.apache.hive.jdbc.HiveDriver`.

To install only the JDBC driver on your Hive clients, proceed as follows.

> **Note:**
>
> The CDH 5.2 Hive JDBC driver is not wire-compatible with the CDH 5.1 version of HiveServer2. Make sure you upgrade Hive clients and all other Hive hosts in tandem: the server first, and then the clients.

1. Install the package (it is included in CDH packaging). Use one of the following commands, depending on the target operating system:
   - On Red-Hat-compatible systems:

```
$ sudo yum install hive-jdbc
```

   - On SLES systems:

```
$ sudo zypper install hive-jdbc
```

   - On Ubuntu or Debian systems:

```
$ sudo apt-get install hive-jdbc
```

2. Add `/usr/lib/hive/lib/*.jar` and `/usr/lib/hadoop/*.jar` to your classpath.

You are now ready to run your JDBC client. HiveServer2 has a new JDBC driver that supports both embedded and remote access to HiveServer2. The connection URLs are also different from those in previous versions of Hive.

For more information see the HiveServer2 Client document.

Connection URLs

The HiveServer2 connection URL has the following format:

```
jdbc:hive2://<host1>:<port1>,<host2>:<port2>/dbName;sess_var_list?hive_conf_list#hive_var_list
```

where:

- *`<host1>:<port1>,<host2>:<port2>`* is a server instance or a comma separated list of server instances to connect to (if dynamic service discovery is enabled). If no server is mentioned here, the embedded server will be used.
- *`dbName`* is the name of the initial database.
- *`sess_var_list`* is a semicolon separated list of key=value pairs of session variables. For example, `user=foo;password=bar`.
- *`hive_conf_list`* is a semicolon separated list of key=value pairs of Hive configuration variables for this session. For example, `hive.server2.transport.mode=http;hive.server2.thrift.http.path=hs2`.
- *`hive_var_list`* is a semicolon separated list of key=value pairs of Hive variables for this session.

Connection URLs for Remote or Embedded Mode: For remote or embedded access, the JDBC Driver class is `org.apache.hive.jdbc.HiveDriver`.

- For a remote server, the URL format is `jdbc:hive2://<host>:<port>/<db>`. The default HiveServer2 port is 10000).
- For an embedded server, the URL format is `jdbc:hive2://` (no host or port).

Connection URLs in HTTP Mode:

```
jdbc:hive2://<host>:<port>/<db>?hive.server2.transport.mode=http;hive.server2.thrift.http.path=<http_endpoint>
```

where `<http_endpoint>` is the corresponding HTTP endpoint configured in `hive-site.xml`. The default value for the endpoint is `cliservice`. The default port for HTTP transport mode is 10001.

Connection URLs with SSL Enabled:

```
jdbc:hive2://<host>:<port>/<db>;ssl=true;sslTrustStore=<trust_store_path>;trustStorePassword=<trust_store_password>
```

where:

- `<trust_store_path>` is the path where client's truststore file is located.
- `<trust_store_password>` is the password to access the truststore.

In HTTP mode with SSL enabled, the URL is of the format:

```
jdbc:hive2://<host>:<port>/<db>;ssl=true;sslTrustStore=<trust_store_path>;trustStorePassword=<trust_store_password>?hive.server2.transport.mode=http;hive.server2.thrift.http.path=<http_endpoint>
```

Setting HADOOP_MAPRED_HOME

- For each user who will be submitting MapReduce jobs using MapReduce v2 (YARN), or running Pig, Hive, or Sqoop in a YARN installation, make sure that the `HADOOP_MAPRED_HOME` environment variable is set correctly, as follows:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
```

- For each user who will be submitting MapReduce jobs using MapReduce v1 (MRv1), or running Pig, Hive, or Sqoop in an MRv1 installation, set the `HADOOP_MAPRED_HOME` environment variable as follows:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-0.20-mapreduce
```

Configuring the Metastore to Use HDFS High Availability

See Configuring Other CDH Components to Use HDFS HA.

Viewing the Hive Documentation

For additional Hive documentation, see the Apache Hive wiki.

To view the Cloudera video tutorial about using Hive, see Introduction to Apache Hive.

Managing Hive

Apache Hive is a powerful data warehousing application for Hadoop. It enables you to access your data using HiveQL, a language similar to SQL.

Hive Roles

Hive is implemented in three roles:

- Hive metastore - Provides metastore services when Hive is configured with a remote metastore.

  Cloudera recommends using a remote Hive metastore, especially for CDH 4.2 or higher. Because the remote metastore is recommended, Cloudera Manager treats the Hive Metastore Server as a required role for all Hive services. A remote metastore provides the following benefits:

  – The Hive metastore database password and JDBC drivers do not need to be shared with every Hive client; only the Hive Metastore Server does. Sharing passwords with many hosts is a security issue.
  – You can control activity on the Hive metastore database. To stop all activity on the database, stop the Hive Metastore Server. This makes it easy to back up and upgrade, which require all Hive activity to stop.

  See Configuring the Hive Metastore (CDH 4) or Configuring the Hive Metastore (CDH 5).

  For information about configuring a remote Hive metastore database with Cloudera Manager, see Cloudera Manager and Managed Service Datastores. To configure high availability for the Hive metastore, see Hive Metastore High Availability on page 68.

- HiveServer2 - Enables remote clients to run Hive queries, and supports a Thrift API tailored for JDBC and ODBC clients, Kerberos authentication, and multi-client concurrency. A CLI named Beeline is also included. See HiveServer2 documentation (CDH 4) or HiveServer2 documentation (CDH 5) for more information.
- WebHCat - HCatalog is a table and storage management layer for Hadoop that makes the same table information available to Hive, Pig, MapReduce, and Sqoop. Table definitions are maintained in the Hive metastore, which HCatalog requires. WebHCat allows you to access HCatalog using an HTTP (REST style) interface.

Hive Execution Engines

Hive in CDH supports two execution engines: MapReduce and Spark. To configure an execution engine perform one of following steps:

- **Beeline** - *(Can be set per query)* Run the `set hive.execution.engine=`*`engine`* command, where *engine* is either `mr` or `spark`. The default is `mr`. For example:

```
set hive.execution.engine=spark;
```

  To determine the current setting, run

```
set hive.execution.engine;
```

- **Cloudera Manager** (*Affects all queries, not recommended*).
  1. Go to the Hive service.
  2. Click the **Configuration** tab.
  3. Search for "execution".
  4. Set the **Default Execution Engine** property to MapReduce or Spark. The default is MapReduce.
  5. Click **Save Changes** to commit the changes.
  6. Return to the Home page by clicking the Cloudera Manager logo.

7. Click ⏻ to invoke the cluster restart wizard.
8. Click **Restart Stale Services**.
9. Click **Restart Now**.
10. Click **Finish**.

Managing Hive Using Cloudera Manager

Cloudera Manager uses the Hive metastore, HiveServer2, and the WebHCat roles to manage the Hive service across your cluster. Using Cloudera Manager, you can configure the Hive metastore, the execution engine (either MapReduce or Spark), and manage HiveServer2.

How Hive Configurations are Propagated to Hive Clients

Because the Hive service does not have worker roles, another mechanism is needed to enable the propagation of client configurations to the other hosts in your cluster. In Cloudera Manager gateway roles fulfill this function. Whether you add a Hive service at installation time or at a later time, ensure that you assign the gateway roles to hosts in the cluster. If you do not have gateway roles, client configurations are not deployed.

Considerations When Upgrading Cloudera Manager

Cloudera Manager 4.5 added support for Hive, which includes the Hive Metastore Server role type. This role manages the metastore process when Hive is configured with a remote metastore.

When upgrading from Cloudera Manager versions before 4.5, Cloudera Manager automatically creates new Hive services to capture the previous implicit Hive dependency from Hue and Impala. Your previous services continue to function without impact. If Hue was using a Hive metastore backed by a Derby database, the newly created Hive Metastore Server also uses Derby. Because Derby does not allow concurrent connections, Hue continues to work, but the new Hive Metastore Server does not run. The failure is harmless (because nothing uses this new Hive Metastore Server at this point) and intentional, to preserve the set of cluster functionality as it was before upgrade. Cloudera discourages the use of a Derby-backed Hive metastore due to its limitations and recommends switching to a different supported database.

Cloudera Manager provides a Hive configuration option to bypass the Hive Metastore Server. When this configuration is enabled, Hive clients, Hue, and Impala connect directly to the Hive metastore database. Prior to Cloudera Manager 4.5, Hue and Impala connected directly to the Hive metastore database, so the bypass mode is enabled by default when upgrading to Cloudera Manager 4.5 and higher. This ensures that the upgrade does not disrupt your existing setup. You should plan to disable the bypass mode, especially when using CDH 4.2 and higher. Using the Hive Metastore Server is the recommended configuration, and the WebHCat Server role requires the Hive Metastore Server to *not* be bypassed. To disable bypass mode, see Disabling Bypass Mode on page 42.

Cloudera Manager 4.5 and higher also supports HiveServer2 with CDH 4.2. In CDH 4, HiveServer2 is not added by default, but can be added as a new role under the Hive service (see Role Instances). In CDH 5, HiveServer2 is a mandatory role.

Disabling Bypass Mode

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

In bypass mode Hive clients directly access the metastore database instead of using the Hive Metastore Server for metastore information.

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Select **Scope** > ***HIVE service_name* (Service-Wide)**
4. Select **Category** > **Advanced**.
5. Clear the **Bypass Hive Metastore Server** property.
6. Click **Save Changes** to commit the changes.

7. Re-deploy Hive client configurations.
8. Restart Hive and any Hue or Impala services configured to use that Hive service.

Running Hive on Spark

This section explains how to run Hive using the Spark execution engine. It assumes that the cluster is managed by Cloudera Manager.

Configuring Hive on Spark

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

To configure Hive to run on Spark do both of the following steps:

- Configure the Hive client to use the Spark execution engine as described in Hive Execution Engines on page 41.
- Identify the Spark service that Hive uses. Cloudera Manager automatically sets this to the configured MapReduce or YARN service and the configured Spark service. See Configuring the Hive Dependency on a Spark Service on page 43.

Configuring the Hive Dependency on a Spark Service

By default, if a Spark service is available, the Hive dependency on the Spark service is configured. To change this configuration, do the following:

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Search for the **Spark On YARN Service**. To configure the Spark service, select the Spark service name. To remove the dependency, select **none**.
4. Click **Save Changes** to commit the changes.
5. Go to the Spark service.
6. Add a Spark gateway role to the host running HiveServer2.
7. Return to the Home page by clicking the Cloudera Manager logo.
8. Click ⏻ to invoke the cluster restart wizard.
9. Click **Restart Stale Services**.
10. Click **Restart Now**.
11. Click **Finish**.
12. In the Hive client, configure the Spark execution engine.

Configuring Hive on Spark for Performance

For the configuration automatically applied by Cloudera Manager when the Hive on Spark service is added to a cluster, see Hive on Spark Autoconfiguration.

For information on configuring Hive on Spark for performance, see Tuning Hive on Spark on page 64.

Troubleshooting Hive on Spark

Delayed result from the first query after starting a new Hive on Spark session

Symptom

The first query after starting a new Hive on Spark session might be delayed due to the start-up time for the Spark on YARN cluster.

Cause

The query waits for YARN containers to initialize.

Solution

No action required. Subsequent queries will be faster.

Exception in HiveServer2 log and HiveServer2 is down

Symptom

In the HiveServer2 log you see the following exception: `Error: org.apache.thrift.transport.TTransportException (state=08S01,code=0)`

Cause

HiveServer2 memory is set too small. For more information, see `stdout` for HiveServer2.

Solution

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Search for Java Heap Size of HiveServer2 in Bytes, and increase the value. Cloudera recommends a minimum value of 2 GB.
4. Click **Save Changes** to commit the changes.
5. Restart HiveServer2.

Out-of-memory error

Symptom

In the log you see an out-of-memory error similar to the following:

```
15/03/19 03:43:17 WARN channel.DefaultChannelPipeline:
An exception was thrown by a user handler while handling an exception event ([id:
0x9e79a9b1, /10.20.118.103:45603 => /10.20.120.116:39110]
      EXCEPTION: java.lang.OutOfMemoryError: Java heap space)
      java.lang.OutOfMemoryError: Java heap space
```

Cause

The Spark driver does not have enough off-heap memory.

Solution

Increase the driver memory `spark.driver.memory` and ensure that `spark.yarn.driver.memoryOverhead` is at least 20% that of the driver memory.

Spark applications stay alive forever

Symptom

Cluster resources are consumed by Spark applications.

Cause

This can occur if you run multiple Hive on Spark sessions concurrently.

Solution

Manually terminate the Hive on Spark applications:

1. Go to the YARN service.
2. Click the **Applications** tab.
3. In the row containing the Hive on Spark application, select ☰ ▾ > **Kill**.

HiveServer2 Web UI

The HiveServer2 web UI provides access to Hive configuration settings, local logs, metrics, and information about active sessions and queries. The HiveServer2 web UI is enabled in newly created clusters running CDH 5.7 and higher, and those using Kerberos are configured for SPNEGO. Clusters upgraded from a previous CDH version must be configured to enable the web UI; see HiveServer2 Web UI Configuration on page 45.

Accessing the HiveServer2 Web UI

Access the HiveServer2 web UI by clicking the **HiveServer2 Web UI** link in Cloudera Manager or by pointing your browser to `http://<host>:<port>/hiveserver2.jsp`.

The following information is displayed:

- **Home** (`/hiveserver2.jsp`): Active sessions, the latest Hive queries, and attributes of the Hive software.
- **Local Logs** (`/logs`): The latest HiverServer2 logs.
- **Metrics Dump** (`/jmx`): Real-time Java Management Extensions (JMX) metrics in JSON format.
- **Hive Configuration** (`/conf`): The current HiveServer2 configuration in XML format.
- **Stack Trace** (`/stacks`): A stack trace of all active threads.

HiveServer2 Web UI Configuration

For managed deployments, configure the HiveServer2 web UI in Cloudera Manager. See Configuring the HiverServer2 Web UI in Cloudera Manager on page 45.

For deployments not managed by Cloudera Manager, edit the configuration file `/etc/hive/conf/hive-site.xml`. To view the HiveServer2 web UI, go to `http://<host>:<port>/hiveserver2.jsp`.

Configurable Properties

HiveServer2 web UI properties, with their default values in Cloudera Hadoop, are:

```
hive.server2.webui.max.threads=50
hive.server2.webui.host=0.0.0.0
hive.server2.webui.port=10002
hive.server2.webui.use.ssl=false
hive.server2.webui.keystore.path=""
hive.server2.webui.keystore.password=""
hive.server2.webui.max.historic.queries=25
hive.server2.webui.use.spnego=false
hive.server2.webui.spnego.keytab=""
hive.server2.webui.spnego.principal=<dynamically sets special string, _HOST, as
hive.server2.webui.host or host name>
```

Tip: To disable the HiveServer2 web UI, set the port to 0 or a negative number

Configuring the HiverServer2 Web UI in Cloudera Manager

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

Note: By default, newly created CDH 5.7 (and higher) clusters have the HiveServer2 web UI enabled, and if using Kerberos, are configured for SPNEGO. Clusters upgraded from an earlier CDH version must have the UI enabled with the port property; other default values can be preserved in most cases.

Configure the HiveServer2 web UI properties in Cloudera Manager on the Configuration tab.

1. Go to the **Hive** service.
2. Click the **Configuration** tab.
3. Select **Scope** > **HiveServer2**.
4. Search for "webui".
5. Locate the properties you want to set and enter your preferred values.

6. Click **Save Changes** to commit the changes.
7. Select **Actions** > **Restart** and when done, click **Close**.
8. Click **HiveServer2 Web UI** to view your changes.

You can use anAdvance Configuration Snippet to set properties that have no dedicated configuration field:

1. On the Hive **Configuration** tab, search for "**HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml**".
2. Click the plus icon to expand configurable attributes for each property.
3. Enter values for **Name**, **Value**, and **Description**.
4. Click the **Final** check box to ensure the value cannot be overwritten.
5. Click **Save Changes** and select **Actions** > **Restart** > **Close**.
6. Click **HiveServer2 Web UI** to view your changes.

Hive Table Statistics

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

Statistics for Hive can be numbers of rows of tables or partitions and the histograms of interesting columns. Statistics are used by the cost functions of the query optimizer to generate query plans for the purpose of query optimization.

If your cluster has Impala then you can use the Impala implementation to compute statistics. The Impala implementation to compute table statistics is available in CDH 5.0.0 or higher and in Impala version 1.2.2 or higher. The Impala implementation of COMPUTE STATS requires no setup steps and is preferred over the Hive implementation. See Overview of Table Statistics. If you are running an older version of Impala, you can collect statistics on a Hive table by running the following command from a Beeline client connected to HiveServer2:

```
analyze table <table name> compute statistics;
analyze table <table name> compute statistics for columns <all columns of a table>;
```

Managing User-Defined Functions (UDFs) with HiveServer2

Hive's query language (HiveQL) can be extended with Java-based user-defined functions (UDFs). See the Apache Hive Language Manual UDF page for information about Hive built-in UDFs. To create customized UDFs, see the Apache Hive wiki. After creating a new Java class to extend the com.example.hive.udf package, you must compile your code into a Java archive file (JAR), and add it to the Hive classpath with the ADD JAR command. The ADD JAR command does *not* work with HiveServer2 and the Beeline client when Beeline runs on a different host. As an alternative to ADD JAR, Hive's *auxiliary paths* functionality should be used.

Perform one of the following procedures depending on whether you want to create permanent or temporary functions.

Blacklist for Built-in UDFs

HiveServer2 maintains a blacklist for built-in UDFs to secure itself against attacks in a multiuser scenario where the hive user's credentials can be used to execute any Java code.

hive.server2.builtin.udf.blacklist	A comma separated list of built-in UDFs that are not allowed to be executed. A UDF that is included in the list will return an error if invoked from a query. Default value: Empty

User-Defined Functions (UDFs) with HiveServer2 Using Cloudera Manager

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

Creating Permanent Functions

1. Copy the JAR file to HDFS and make sure the hive user can access this JAR file.
2. Copy the JAR file to the host on which HiveServer2 is running. Save the JARs to any directory you choose, give the hive user read, write, and execute access to this directory, and make a note of the path (for example, /opt/local/hive/lib/).

> **Note:** If the Hive Metastore is running on a different host, create the same directory there that you created on the HiveServer2 host. You do not need to copy the JAR file onto the Hive Metastore host, but the same directory must be there. For example, if you copied the JAR file to /opt/local/hive/lib/ on the HiveServer2 host, you must create the same directory on the Hive Metastore host. If the same directory is not present on the Hive Metastore host, Hive Metastore service will not start.

3. In the Cloudera Manager Admin Console, go to the Hive service.
4. Click the **Configuration** tab.
5. Expand the **Hive (Service-Wide)** scope.
6. Click the **Advanced** category.
7. Configure the **Hive Auxiliary JARs Directory** property with the HiveServer2 host path and the Hive Metastore host path from Step 2, for example /opt/local/hive/lib/. Setting this property overwrites hive.aux.jars.path, even if this variable has been previously set in the HiveServer2 advanced configuration snippet.
8. Click **Save Changes**. The JARs are added to HIVE_AUX_JARS_PATH environment variable.
9. Redeploy the Hive client configuration.

   a. In the Cloudera Manager Admin Console, go to the Hive service.
   b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
   c. Click **Deploy Client Configuration**.

10. Restart the Hive service.
11. ***With Sentry enabled*** - Grant privileges on the JAR files to the roles that require access. Log in to Beeline as user hive and use the Hive SQL GRANT statement to do so. For example:

```
GRANT ALL ON URI 'file:///opt/local/hive/lib/my.jar' TO ROLE EXAMPLE_ROLE
```

12. Run the CREATE FUNCTION command to create the UDF from the JAR file.

    With Sentry enabled - On a Sentry secured cluster, the USING JAR command is not supported. To load the jar, you have to make sure the JAR file is at the location pointed to by either hive.aux.jars.path or hive.reloadable.aux.jars.path.

```
CREATE FUNCTION addfunc AS 'com.example.hiveserver2.udf.add';
```

Without Sentry - Run the CREATE FUNCTION command as follows and point to the JAR file location in HDFS. For example:

```
CREATE FUNCTION addfunc AS 'com.example.hiveserver2.udf.add' USING JAR
'hdfs:///path/to/jar'
```

Creating Temporary Functions

1. Copy the JAR file to the host on which HiveServer2 is running. Save the JARs to any directory you choose, give the hive user read, write, and execute access to this directory, and make a note of the path (for example, /opt/local/hive/lib/).

> **Note:** If the Hive Metastore is running on a different host, create the same directory there that you created on the HiveServer2 host. You do not need to copy the JAR file onto the Hive Metastore host, but the same directory must be there. For example, if you copied the JAR file to `/opt/local/hive/lib/` on the HiveServer2 host, you must create the same directory on the Hive Metastore host. If the same directory is not present on the Hive Metastore host, Hive Metastore service will not start.

2. In the Cloudera Manager Admin Console, go to the Hive service.
3. Click the **Configuration** tab.
4. Expand the **Hive (Service-Wide)** scope.
5. Click the **Advanced** category.
6. Configure the **Hive Auxiliary JARs Directory** property with the HiveServer2 host path and the Hive Metastore host path from Step 1, for example `/opt/local/hive/lib/`. Setting this property overwrites `hive.aux.jars.path`, even if this variable has been previously set in the HiveServer2 advanced configuration snippet.
7. Click **Save Changes**. The JARs are added to `HIVE_AUX_JARS_PATH` environment variable.
8. Redeploy the Hive client configuration.

   a. In the Cloudera Manager Admin Console, go to the Hive service.
   b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
   c. Click **Deploy Client Configuration**.

9. Restart the Hive service.
10. Run the `CREATE TEMPORARY FUNCTION` command. For example:

```
CREATE TEMPORARY FUNCTION addfunc AS 'com.example.hiveserver2.udf.add'
```

User-Defined Functions (UDFs) with HiveServer2 Using the Command Line

The following sections describe how to create permanent and temporary functions using the command line.

Creating Permanent Functions

1. Copy the JAR file to HDFS and make sure the `hive` user can access this jar file.
2. On the Beeline client machine, in `/etc/hive/conf/hive-site.xml`, set the `hive.aux.jars.path` property to a comma-separated list of the fully qualified paths to the JAR file and any dependent libraries.

```
hive.aux.jars.path=file:///opt/local/hive/lib/my.jar
```

3. Copy the JAR file (and its dependent libraries) to the host running HiveServer2/Impala. Make sure the `hive` user has read, write, and execute access to these files on the HiveServer2/Impala host.
4. On the HiveServer2/Impala host, open `/etc/default/hive-server2` and set the `AUX_CLASSPATH` variable to a comma-separated list of the fully qualified paths to the JAR file and any dependent libraries.

```
AUX_CLASSPATH=/opt/local/hive/lib/my.jar
```

5. Restart HiveServer2.
6. ***With Sentry enabled*** - Grant privileges on the JAR files to the roles that require access. Login to Beeline as user `hive` and use the Hive SQL GRANT statement to do so. For example:

```
GRANT ALL ON URI 'file:///opt/local/hive/lib/my.jar' TO ROLE EXAMPLE_ROLE
```

   If you are using Sentry policy files, you can grant the URI privilege as follows:

```
udf_r = server=server1->uri=file:///opt/local/hive/lib
udf_r = server=server1->uri=hdfs:///path/to/jar
```

7. Run the CREATE FUNCTION command to create the UDF from the JAR file.

   With Sentry enabled - On a Sentry secured cluster, the USING JAR command is not supported. To load the jar, you have to make sure the JAR file is at the location pointed to by either hive.aux.jars.path or hive.reloadable.aux.jars.path.

```
CREATE FUNCTION addfunc AS 'com.example.hiveserver2.udf.add';
```

   Without Sentry - Run the CREATE FUNCTION command as follows and point to the JAR file location in HDFS. For example:

```
CREATE FUNCTION addfunc AS 'com.example.hiveserver2.udf.add' USING JAR
'hdfs:///path/to/jar'
```

Creating Temporary Functions

1. On the Beeline client machine, in /etc/hive/conf/hive-site.xml, set the hive.aux.jars.path property to a comma-separated list of the fully qualified paths to the JAR file and any dependent libraries.

```
hive.aux.jars.path=file:///opt/local/hive/lib/my.jar
```

2. Copy the JAR file (and its dependent libraries) to the host running HiveServer2/Impala. Make sure the hive user has read, write, and execute access to these files on the HiveServer2/Impala host.
3. On the HiveServer2/Impala host, open /etc/default/hive-server2 and set the AUX_CLASSPATH variable to a comma-separated list of the fully qualified paths to the JAR file and any dependent libraries.

```
AUX_CLASSPATH=/opt/local/hive/lib/my.jar
```

4. Restart HiveServer2.
5. Run the CREATE TEMPORARY FUNCTION command and point to the JAR from Hive:

```
CREATE TEMPORARY FUNCTION addfunc AS 'com.example.hiveserver2.udf.add'
```

Update Existing HiveServer2 User-Defined Functions (UDFs)

Consider a sample UDF, hive_udf.jar that is already in use. If you want to update the sample function, my_udf, in the hive_udf.jar JAR file, you first drop my_udf using the DROP FUNCTION command. Then, delete the old JAR file and place the new JAR with the updated function in both, HDFS and the local filesystem. Finally, re-create the my_udf function defined in the new JAR using the CREATE FUNCTION command. These steps have been described in detail below:

1. Drop the function to be updated, in this case, my_udf. This can be done by using bash scripts to first, connect to HiveServer2 using Beeline, and then, drop the function. First create an .hql file with the DROP FUNCTION command:

```
# more drop_func.hql

drop function my_udf;
```

   Create a bash script to connect to HiveServer2 using Beeline and the previously-created drop_func.hql:

```
# more drop_func.sh

beeline -u
"jdbc:hive2://hiveServer2:10000/default;principal=hive/HiveServer2@CLOUDERA.COM" -f
drop_func.hql
```

   Run the bash script:

```
# ./drop_func.sh
```

2. Delete the old JAR file, `hive_udf.jar` from both HDFS, and the local filesystem. On the local filesystem, the JAR can be found at the location pointed to by either `hive.aux.jars.path` or `hive.reloadable.aux.jars.path`.
3. Place the updated `hive_udf.jar` in HDFS and the local filesystem (for example, `/user/hive/udf`). This time, make sure you add the JAR to the path set by `hive.reloadable.aux.jars.path`. You do not need to place the JAR file at the path pointed to by `hive.aux.jars.path`.

   If you are using Cloudera Manager, use an advanced configuration snippet to set the value for `hive.reloadable.aux.jars.path`.

   1. In the Cloudera Manager Admin Console, go to the Hive service.
   2. Click the **Configuration** tab.
   3. Expand the **Hive (Service-Wide)** > **Advanced** categories.
   4. Locate the **HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml** property and add:

```
<property>
    <name>hive.reloadable.aux.jars.path</name>
    <value>/user/hive/udf</value>
</property>
```

   5. Delete the value for the **Hive Auxiliary JARs Directory** property.
   6. Click **Save Changes** to commit the changes.

4. Recreate functions defined in `hive-udf.jar`. This can also be done by using bash scripts to connect to HiveServer2 and run the `CREATE FUNCTION` command. First create an .hql file with the `CREATE FUNCTION` command:

```
 # more create_func.hql

CREATE FUNCTION my_udf as ‘com.cloudera.sa.hiveudf.myudf’;
```

   Create a bash script to connect to HiveServer2 using Beeline and the previously-created `create_func.hql`:

```
# more create_func.sh

beeline -u
"jdbc:hive2://hiveServer2:10000/default;principal=hive/HiveServer2@CLOUDERA.COM" -f
create_func.hql
```

   Then run the bash script:

```
# ./create_func.sh
```

5. If you already have a Beeline session open before the JAR was updated, run the RELOAD command. The new UDF will take effect once the command completes successfully.

Configuring Transient Hive ETL Jobs to Use the Amazon S3 Filesystem

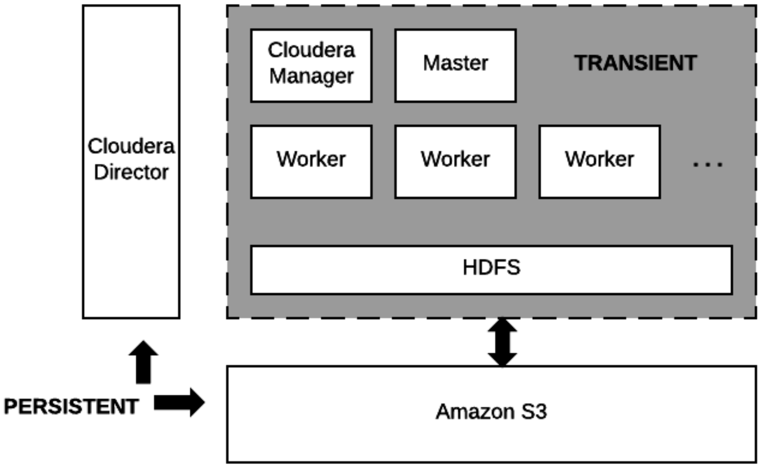
Hive is a popular choice for batch extract-transform-load (ETL) jobs such as cleaning, serializing, deserializing, and transforming data. In on-premise deployments, ETL jobs operate on data stored in a permanent Hadoop cluster that runs HDFS on local disks. However, ETL jobs are frequently transient and can benefit from cloud deployments where cluster nodes can be quickly created and torn down as needed. This approach can translate to significant cost savings.

> **Important:** Cloudera components writing data to S3 are constrained by the inherent limitation of Amazon S3 known as "eventual consistency." For more information, see Data Storage Considerations.

About Transient Jobs

Most ETL jobs on transient clusters run from scripts that make API calls to a provisioning service such as Cloudera Director. They can be triggered by external events, such as IoT (internet of things) events like reaching a temperature threshold, or they can be run on a regular schedule, such as every day at midnight.

Transient Jobs Hosted on Amazon S3



Data residing on Amazon S3 and the node running Cloudera Director are the only persistent components. The computing nodes and local storage come and go with each transient workload.

Configuring and Running Jobs on Transient Clusters

Using AWS to run transient jobs involves the following steps, which are documented in an end-to-end example you can download from this Cloudera GitHub repository. Use this example to test transient clusters with Cloudera Director.

1. Configure AWS settings.
2. Install Cloudera Director server and client.
3. Design and test a cluster configuration file for the job.
4. Prepare Amazon Machine Images (AMIs) with preloaded and pre-extracted CDH parcels.
5. Package the job into a shell script with the necessary bootstrap steps.
6. Prepare a job submission script.
7. Schedule the recurring job.

See Optimizing Hive Write Performance on Amazon S3 for information about tuning Hive to write data to S3 tables or partitions.

Configuring AWS Settings

Use the AWS web console to configure Virtual Private Clouds (VPCs), Security Groups, and Identity and Access Management (IAM) roles on AWS before you install Cloudera Director.

Best Practices

Networking

Cloudera recommends deploying clusters within a VPC, using Security Groups to control network traffic. Each cluster should have outbound internet connectivity through a NAT (network address translation) server when you deploy in a private subnet. If you deploy in a public subnet, each cluster needs direct connectivity. Inbound connections should be limited to traffic from private IPs within the VPC and SSH access through port 22 to the gateway nodes from approved IP addresses. For details about using Cloudera Director to perform these steps, see Setting up the AWS Environment.

Data Access

Create an IAM role that gives the cluster access to S3 buckets. Using IAM roles is a more secure way to provide access to S3 than adding the S3 keys to Cloudera Manager by configuring `core-site.xml` safety valves.

AWS Placement Groups

To improve performance, place worker nodes in an AWS *placement group*. See Placement Groups in the AWS documentation set.

Install Cloudera Director

See Launching an EC2 Instance for Cloudera Director. Install Cloudera Director server and client in a virtual machine that can reach the VPC you set up in the Configuring AWS section.

Create the Cluster Configuration File

The cluster configuration file contains the information that Director needs to bootstrap and properly configure a cluster:

- Deployment environment configuration.
- Instance groups configuration.
- List of services.
- Pre- and post-creation scripts.
- Custom service and role configurations.
- Billing ID and license for hourly billing for Director use from Cloudera. See Usage-Based Billing.

Creating the cluster configuration file represents the bulk of the work of configuring Hive to use the S3 filesystem. This GitHub repository contains sample configurations for different cloud providers.

Testing the Cluster Configuration File

After defining the cluster configuration file, test it to make sure it runs without errors:

1. Use secure shell (SSH) to log in to the server running Cloudera Director.
2. Run the `validate` command by passing the configuration file to it:

```
cloudera-director validate <cluster_configuration_file_name.conf>
```

   If Cloudera Director server is running in a separate instance from the Cloudera Director client, you must run:

```
cloudera-director bootstrap-remote <admin_username> --lp.remote.password=<admin_password>
 --lp.remote.hostAndPort=<host_name>:<port_number>
```

Prepare the CDH AMIs

It is not a requirement to have preloaded AMIs containing CDH parcels that are already extracted. However, preloaded AMIs significantly speed up the cluster provisioning process. See this repo in GitHub for instructions and scripts that create preloaded AMIs.

After you have created preloaded AMIs, replace the AMI IDs in the cluster configuration file with the new preloaded AMI IDs to ensure that all cluster instances use the preloaded AMIs.

Add the following code to the cluster configuration file in the `cloudera-manager` section to make sure that Cloudera Manager does not try to download the parcels:

```
cloudera-manager {
...
     configs {
          CLOUDERA_MANAGER {
               customer_banner_html: "Managed by Cloudera Director"
                    MANAGES_PARCELS: false
          }
     }
...
}
```

Run the Cloudera Director validate command again to test bringing up the cluster. See Testing the Cluster Configuration File. The cluster should come up significantly faster than it did when you tested it before.

Prepare the Job Wrapper Script

Define the Hive query or job that you want to execute and a wrapper shell script that runs required prerequisite commands before it executes the query or job on the transient cluster. The Director public GitHub repository contains simple examples of a MapReduce job wrapper script and an Oozie job wrapper script.

For example, the following is a Bash shell wrapper script for a Hive query:

```
set -x -e
sudo -u hdfs hadoop fs -mkdir /user/ec2-user
sudo -u hdfs hadoop fs -chown ec2-user:ec2-user /user/ec2-user
hive -f query.q
exit 0
```

Where query.q contains the Hive query. After you create the job wrapper script, test it to make sure it runs without errors.

Log Collection

Save all relevant log files in S3 because they disappear when you terminate the transient cluster. Use these log files to debug any failed jobs after the cluster is terminated. To save the log files, add an additional step to your job wrapper shell script.

Example for copying Hive logs from a transient cluster node to S3:

```
# Install AWS CLI
curl "https://s3.amazonaws.com/aws-cli/awscli-bundleszip" -o "awscli-bundle.zip"
sudo yum install -y unzip
unzip awscli-bundle.zip
sudo ./awscli-bundle/install -i /usr/local/aws -b /usr/local/bin/aws

# Set Credentials
export AWS_ACCESS_KEY_ID=[]
export AWS_SECRET_ACCESS_KEY=[]

# Copy Log Files
aws s3 cp /tmp/ec2-user/hive.log s3://bucket-name/output/hive/logs/
```

Prepare the End-to-End Job Submission Script

This script automates the end-to-end workflow, including the following steps:

1. Submit the transient cluster configuration file to Cloudera Director.
2. Wait for the cluster to be provisioned and ready to use.
3. Copy all job-related files to the cluster.
4. Submit the job script to the cluster.
5. Wait for the job to complete.
6. Shutdown the cluster.

See the Cloudera Engineering Blog post How-to: Integrate Cloudera Director with a Data Pipeline in the Cloud for information about creating an end-to-end job submission script. A sample script can be downloaded from GitHub here.

Schedule the Recurring Job

To schedule the recurring job, wrap the end-to-end job submission script in a Cron job or by triggering the script to run when a particular event occurs.

Tuning Hive

To maximize performance of your Apache Hive query workloads, you need to optimize cluster configurations, queries, and underlying Hive table design. This includes the following:

- Configure CDH clusters for the maximum allowed heap memory size, load-balance concurrent connections across your CDH Hive components, and allocate adequate memory to support HiveServer2 and Hive metastore operations.
- Review your Hive query workloads to make sure queries are not overly complex, that they do not access large numbers of Hive table partitions, or that they force the system to materialize all columns of accessed Hive tables when only a subset is necessary.
- Review the underlying Hive table design, which is crucial to maximizing the throughput of Hive query workloads. Do not create thousands of table partitions that might cause queries containing JOINs to overtax HiveServer2 and the Hive metastore. Limit column width, and keep the number of columns under 1,000.

The following sections provide details on implementing these best practices to maximize performance for deployments of HiveServer2 and the Hive metastore.

Heap Size and Garbage Collection for Hive Components

This section provides guidelines for setting HiveServer2 and Hive metastore memory and garbage-collection properties.

Memory Recommendations

HiveServer2 and the Hive metastore require sufficient memory to run correctly. The default heap size of 256 MB for each component is inadequate for production workloads. Consider the following guidelines for sizing the heap for each component, based on your cluster size.

Number of Concurrent Connections	HiveServer2 Heap Size Recommended Range	Hive Metastore Heap Size Recommended Range
Up to 40 concurrent connections Cloudera recommends splitting HiveServer2 into multiple instances and load-balancing once you start allocating over 16 GB to HiveServer2. This reduces the impact of Java garbage collection on active processing by the service.	12 - 16 GB	12 - 16 GB
Up to 20 concurrent connections	6 - 12 GB	10 - 12 GB
Up to 10 concurrent connections	4 - 6 GB	4 - 10 GB
One connection	4 GB	4 GB

Important: These numbers are general guidance only, and can be affected by factors such as number of columns, partitions, complex joins, and client activity. Based on your anticipated deployment, refine through testing to arrive at the best values for your environment.

In addition, the Beeline CLI should use a heap size of at least 2 GB.

Set the PermGen space for Java garbage collection to 512 MB for all.

Configuring Heap Size and Garbage Collection

Using Cloudera Manager

To configure heap size and garbage collection for HiveServer2:

1. To set heap size, go to **Home > Hive > Configuration > HiveServer2 > Resource Management**.
2. Set **Java Heap Size of HiveServer2 in Bytes** to the desired value, and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > HiveServer2 > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M`, the type of garbage collector used (`ConcMarkSweepGC` or `ParNewGC`), and enable or disable the garbage collection overhead limit in **Java Configuration Options for HiveServer2**.

   The following example sets the PermGen space to `512M`, uses the new Parallel Collector, and disables the garbage collection overhead limit:

```
-XX:MaxPermSize=512M -XX:+UseParNewGC -XX:-UseGCOverheadLimit
```

5. From the **Actions** drop-down menu, select `Restart` to restart the HiveServer2 service.

To configure heap size and garbage collection for the Hive metastore:

1. To set heap size, go to **Home > Hive > Configuration > Hive Metastore > Resource Management**.
2. Set **Java Heap Size of Hive Metastore Server in Bytes** to the desired value, and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > Hive Metastore Server > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M`, the type of garbage collector used (`ConcMarkSweepGC` or `ParNewGC`), and enable or disable the garbage collection overhead limit in **Java Configuration Options for Hive Metastore Server**. For an example of this setting, see step 4 above for configuring garbage collection for HiveServer2.
5. From the **Actions** drop-down menu, select `Restart` to restart the Hive Metastore service.

To configure heap size and garbage collection for the Beeline CLI:

1. To set heap size, go to **Home > Hive > Configuration > Gateway > Resource Management**.
2. Set **Client Java Heap Size in Bytes** to at least `2 GiB` and click **Save Changes**.
3. To set garbage collection, go to **Home > Hive > Configuration > Gateway > Advanced**.
4. Set the PermGen space for Java garbage collection to `512M` in **Client Java Configuration Options**.

   The following example sets the PermGen space to `512M` and specifies `IPv4`:

```
-XX:MaxPermSize=512M -Djava.net.preferIPv4Stack=true
```

5. From the **Actions** drop-down menu, select `Restart` to restart the client service.

Using the Command Line

To configure the heap size for HiveServer2 and Hive metastore, set the `-Xmx` parameter in the `HADOOP_OPTS` variable to the desired maximum heap size in `/etc/hive/hive-env.sh`.

To configure the heap size for the Beeline CLI, set the `HADOOP_HEAPSIZE` environment variable in `/etc/hive/hive-env.sh` before starting the Beeline CLI.

The following example shows a configuration with the following settings:

- HiveServer2 uses 12 GB heap.
- Hive metastore uses 12 GB heap.
- Hive clients use 2 GB heap.

The settings to change are in bold. All of these lines are commented out (prefixed with a # character) by default.

```
if [ "$SERVICE" = "cli" ]; then
  if [ -z "$DEBUG" ]; then
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms12288m
              -XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:+UseParNewGC
-XX:-UseGCOverheadLimit"
  else
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms12288m
              -XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:-UseGCOverheadLimit"
  fi
fi

export HADOOP_HEAPSIZE=2048
```

You can use either the Concurrent Collector or the new Parallel Collector for garbage collection by passing `-XX:+useConcMarkSweepGC` or `-XX:+useParNewGC` in the HADOOP_OPTS lines above. To enable the garbage collection overhead limit, remove the `-XX:-UseGCOverheadLimit` setting or change it to `-XX:+UseGCOverheadLimit`.

Set the PermGen space for Java garbage collection to `512M` for all in the `JAVA-OPTS` environment variable. For example:

```

set JAVA_OPTS="-Xms256m -Xmx1024m -XX:PermSize=512m -XX:MaxPermSize=512m"
```

HiveServer2 Performance Tuning and Troubleshooting

HiveServer2 (HS2) services might require more memory if there are:

- Many Hive table partitions.
- Many concurrent connections to HS2.
- Complex Hive queries that access significant numbers of table partitions.

If any of these conditions exist, Hive can run slowly or possibly crash because the entire HS2 heap memory is full. This section describes the symptoms that occur when HS2 needs additional memory, how you can troubleshoot issues to identify their causes, and then address them.

Symptoms Displayed When HiveServer2 Heap Memory is Full

When HS2 heap memory is full, you might experience the following issues:

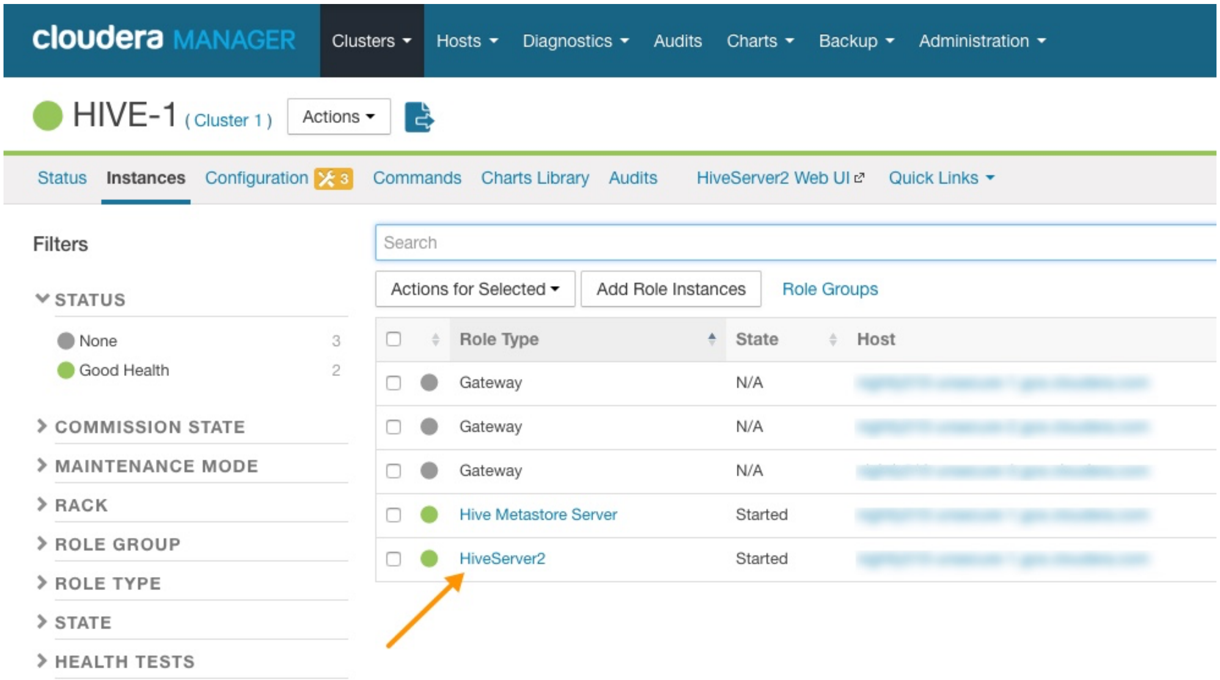
- HS2 service goes down and new sessions fail to start.
- HS2 service seems to be running fine, but client connections are refused.
- Query submission fails repeatedly.
- HS2 performance degrades and displays the following behavior:
  - Query submission delays
  - Long query execution times

Troubleshooting

HiveServer2 Service Crashes

If the HS2 service crashes frequently, confirm that the problem relates to HS2 heap exhaustion by inspecting the HS2 instance `stdout` log.

1. In Cloudera Manager, from the home page, go to **Hive > Instances**.
2. In the Instances page, click the link of the HS2 node that is down:



cloudera MANAGER   Clusters ▾   Hosts ▾   Diagnostics ▾   Audits   Charts ▾   Backup ▾   Administration ▾

● HIVE-1 (Cluster 1)   Actions ▾

Status   Instances   Configuration 3   Commands   Charts Library   Audits   HiveServer2 Web UI   Quick Links ▾

Filters

STATUS
- None 3
- Good Health 2

COMMISSION STATE
MAINTENANCE MODE
RACK
ROLE GROUP
ROLE TYPE
STATE
HEALTH TESTS

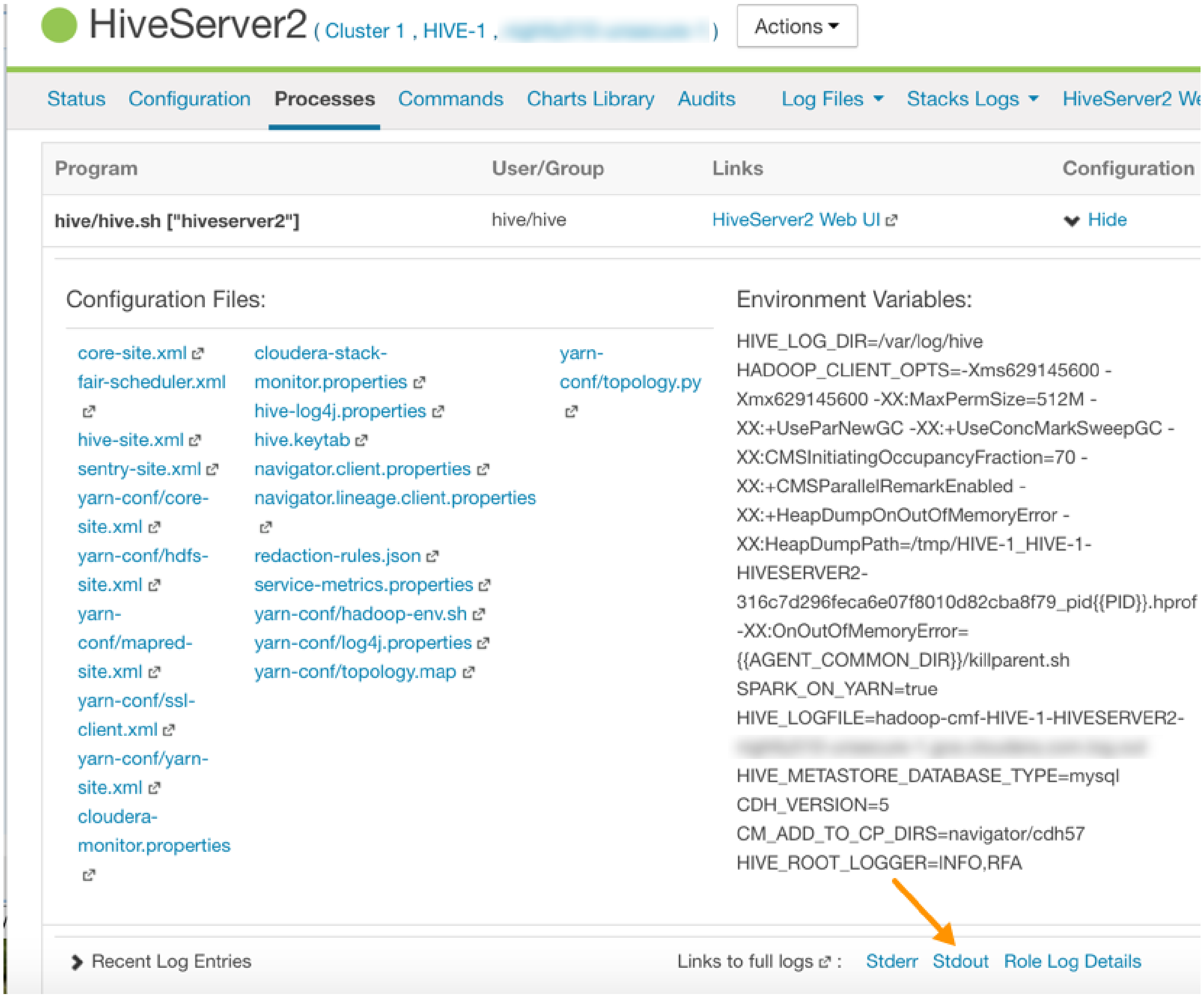
Search

Actions for Selected ▾   Add Role Instances   Role Groups

□		Role Type	State	Host
□	●	Gateway	N/A	[redacted]
□	●	Gateway	N/A	[redacted]
□	●	Gateway	N/A	[redacted]
□	●	Hive Metastore Server	Started	[redacted]
□	●	HiveServer2	Started	[redacted]

Figure 2: HiveServer2 Link on the Cloudera Manager Instances Page

3. On the HiveServer2 page, click **Processes**.
4. On the HiveServer2 Processes page, scroll down to the **Recent Log Entries** and click the link to the **Stdout** log.



Program	User/Group	Links	Configuration
hive/hive.sh ["hiveserver2"]	hive/hive	HiveServer2 Web UI	Hide

Configuration Files:

- core-site.xml
- fair-scheduler.xml
- hive-site.xml
- sentry-site.xml
- yarn-conf/core-site.xml
- yarn-conf/hdfs-site.xml
- yarn-conf/mapred-site.xml
- yarn-conf/ssl-client.xml
- yarn-conf/yarn-site.xml
- cloudera-monitor.properties
- cloudera-stack-monitor.properties
- hive-log4j.properties
- hive.keytab
- navigator.client.properties
- navigator.lineage.client.properties
- redaction-rules.json
- service-metrics.properties
- yarn-conf/hadoop-env.sh
- yarn-conf/log4j.properties
- yarn-conf/topology.map
- yarn-conf/topology.py

Environment Variables:

```
HIVE_LOG_DIR=/var/log/hive
HADOOP_CLIENT_OPTS=-Xms629145600 -
Xmx629145600 -XX:MaxPermSize=512M -
XX:+UseParNewGC -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=70 -
XX:+CMSParallelRemarkEnabled -
XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/tmp/HIVE-1_HIVE-1-
HIVESERVER2-
316c7d296feca6e07f8010d82cba8f79_pid{{PID}}.hprof
-XX:OnOutOfMemoryError=
{{AGENT_COMMON_DIR}}/killparent.sh
SPARK_ON_YARN=true
HIVE_LOGFILE=hadoop-cmf-HIVE-1-HIVESERVER2-
[redacted]
HIVE_METASTORE_DATABASE_TYPE=mysql
CDH_VERSION=5
CM_ADD_TO_CP_DIRS=navigator/cdh57
HIVE_ROOT_LOGGER=INFO,RFA
```

Figure 3: Link to the Stdout Log on the Cloudera Manager Processes Page

5. In the `stdout.log`, look for the following error:

```
# java.lang.OutOfMemoryError: Java heap space
# -XX:OnOutOfMemoryError="/usr/lib64/cmf/service/common/killparent.sh"
# Executing /bin/sh -c "/usr/lib64/cmf/service/common/killparent.sh"
```

Video: Troubleshooting HiveServer2 Service Crashes

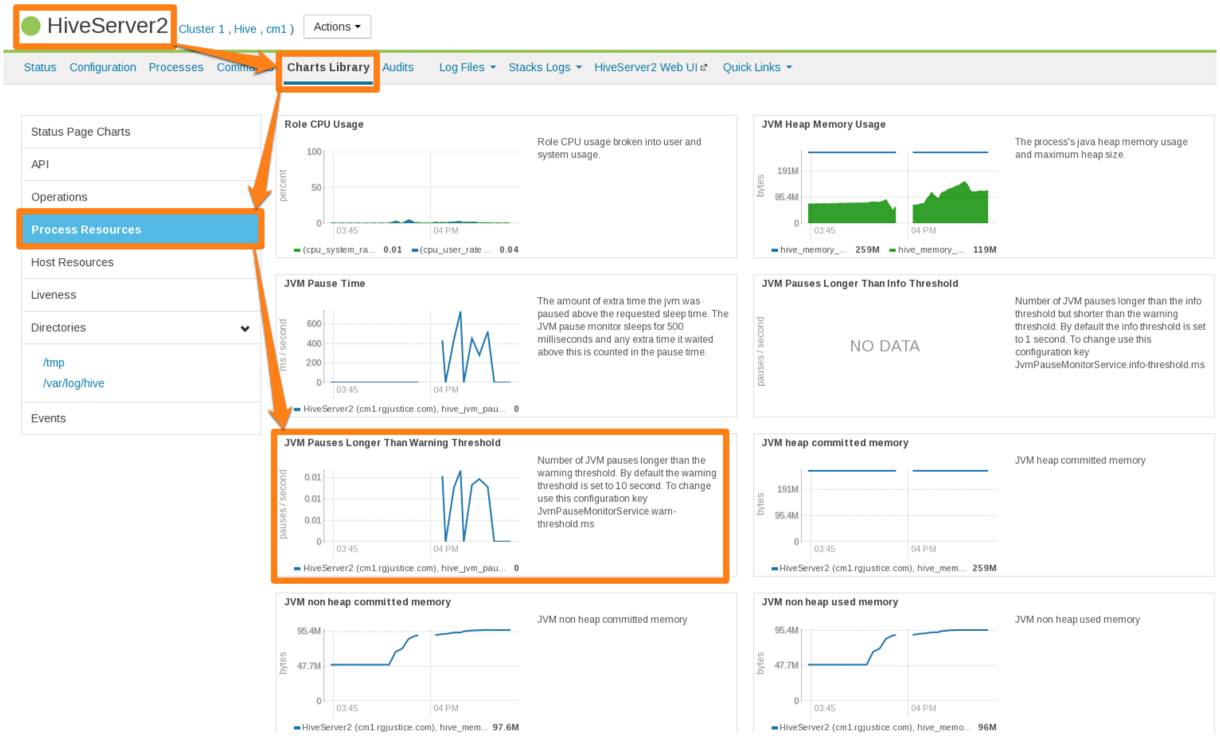
For more information about configuring Java heap size for HiveServer2, see the following video:

After you start the video, click **YouTube** in the lower right corner of the player window to watch it on YouTube where you can resize it for clearer viewing.

HiveServer2 General Performance Problems or Connections Refused

For general HS2 performance problems or if the service refuses connections, but does not completely hang, inspect the Cloudera Manager process charts:

1. In Cloudera Manager, navigate to **Home > Hive > Instances > HiveServer2 > Charts Library**.
2. In the **Process Resources** section of the Charts Library page, view the **JVM Pause Time** and the **JVM Pauses Longer Than Warning Threshold** charts for signs that JVM has paused to manage resources. For example:



Role CPU Usage

Role CPU usage broken into user and system usage.

JVM Heap Memory Usage

The process's java heap memory usage and maximum heap size.

JVM Pause Time

The amount of extra time the jvm was paused above the requested sleep time. The JVM pause monitor sleeps for 500 milliseconds and any extra time it waited above this is counted in the pause time.

JVM Pauses Longer Than Info Threshold

Number of JVM pauses longer than the info threshold but shorter than the warning threshold. By default the info threshold is set to 1 second. To change use this configuration key JvmPauseMonitorService.info-threshold.ms

JVM Pauses Longer Than Warning Threshold

Number of JVM pauses longer than the warning threshold. By default the warning threshold is set to 10 second. To change use this configuration key JvmPauseMonitorService.warn-threshold.ms

JVM heap committed memory

JVM heap committed memory

JVM non heap committed memory

JVM non heap committed memory

JVM non heap used memory

JVM non heap used memory

Figure 4: Cloudera Manager Chart Library Page for Process Resources

HiveServer2 Performance Best Practices

High heap usage by the HS2 process can be caused by Hive queries accessing high numbers of table partitions (greater than several thousand), high levels of concurrency, or other Hive workload characteristics described in Identify Workload Characteristics That Increase Memory Pressure on page 60.

HiveServer2 Heap Size Configuration Best Practices

Optimal HS2 heap size configuration depends on several factors, including workload characteristics, number of concurrent clients, and the partitioning of underlying Hive tables. To resolve HS2 memory-related issues, confirm that the HS2 heap size is set properly for your environment.

1. In CDH 5.7 and higher, Cloudera Manager starts the HS2 service with 4 GB heap size by default unless hosts have insufficient memory. However, the heap size on lower versions of CDH or upgraded clusters might not be set to this recommended value. To raise the heap size to at least 4 GB:

   a. In Cloudera Manager, go to **Home > Hive > Configuration > HiveServer2 > Resource Management**.
   b. Set **Java Heap Size of HiveServer2 in Bytes** to `4 GiB` and click **Save Changes**.
   c. From the **Actions** drop-down menu, select `Restart` to restart the HS2 service.

   If HS2 is already configured to run with 4 GB or greater heap size and there are still performance issues, workload characteristics may be causing memory pressure. Increase heap size to reduce memory pressure on HS2. Cloudera does not recommend exceeding 16 GB per instance because of long garbage collection pause times. See Identify Workload Characteristics That Increase Memory Pressure on page 60 for tips to optimize query workloads to reduce the memory requirements on HS2. Cloudera recommends splitting HS2 into multiple instances and load-balancing once you start allocating over 16 GB to HS2.

2. If workload analysis does not reveal any major issues, or you can only address workload issues over time, consider the following options:

   - Increase the heap size on HS2 in incremental steps. Cloudera recommends increasing the heap size by 50% from the current value with each step. If you have increased the heap size to 16 GB and issues persist, contact Cloudera Support.

- Reduce the number of services running on the HS2 host.
- Load-balance workloads across multiple HS2 instances as described in How the Number of Concurrent Connections Affect HiveServer2 Performance on page 60.
- Add more physical memory to the host or upgrade to a larger server.

How the Number of Concurrent Connections Affect HiveServer2 Performance

The number of concurrent connections can impact HS2 in the following ways:

- **High number of concurrent queries**

  High numbers of concurrent queries increases the connection count. Each query connection consumes resources for the query plan, number of table partitions accessed, and partial result sets. Limiting the number of concurrent users can help reduce overall HS2 resource consumption, especially limiting scenarios where one or more "in-flight" queries returns large result sets.

  How to resolve:

  – Load-balance workloads across multiple HS2 instances by using HS2 load balancing, which is available in CDH 5.7 and later. Cloudera recommends that you determine the total number of HS2 servers on a cluster by dividing the expected maximum number of concurrent users on a cluster by 40. For example, if 400 concurrent users are expected, 10 HS2 instances should be available to support them. See Configuring HiveServer2 High Availability in CDH on page 71 for setup instructions.

  – Review usage patterns, such as batch jobs timing or Oozie workflows, to identify spikes in the number of connections that can be spread over time.

- **Many abandoned Hue sessions**

  Users opening numerous browser tabs in Hue causes multiple sessions and connections. In turn, all of these open connections lead to multiple operations and multiple result sets held in memory for queries that finish processing. Eventually, this situation leads to a resource crisis.

  How to resolve:

  – Reduce the session timeout duration for HS2, which minimizes the impact of abandoned Hue sessions. To reduce session timeout duration, modify these configuration parameters as follows:

    – `hive.server2.idle.operation.timeout=7200000`

      The default setting for this parameter is `21600000` or 6 hours.

    – `hive.server2.idle.session.timeout=21600000`

      The default setting for this parameter is `43200000` or 12 hours.

      To set these parameters in Cloudera Manager, go to **Home > Hive > Configuration > HiveServer2 > Advanced** , and then search for each parameter.

  – Reduce the size of the result set returned by adding filters to queries. This minimizes memory pressure caused by "dangling" sessions.

Identify Workload Characteristics That Increase Memory Pressure

If increasing the heap size based on configuration guidelines does not improve performance, analyze your query workloads to identify characteristics that increase memory pressure on HS2. Workloads with the following characteristics increase memory requirements for HS2:

- **Queries that access a large number of table partitions:**

  – Cloudera recommends that a single query access no more than 10,000 table partitions. If joins are also used in the query, calculate the combined partition count accessed across all tables.

  - Look for queries that load all table partitions in memory to execute. This can substantially add to memory pressure. For example, a query that accesses a partitioned table with the following SELECT statement loads all partitions of the target table to execute:

```
SELECT * FROM <table_name> LIMIT 10;
```

    How to resolve:

    - Add partition filters to queries to reduce the total number of partitions that are accessed. To view all of the partitions processed by a query, run the EXPLAIN DEPENDENCY clause, which is explained in the Apache Hive Language Manual.
    - Set the `hive.metastore.limit.partition.request` parameter to 1000 to limit the maximum number of partitions accessed from a single table in a query. See the Apache wiki for information about setting this parameter. If this parameter is set, queries that access more than 1000 partitions fail with the following error:

```
MetaException: Number of partitions scanned (=%d) on table '%s' exceeds limit (=%d)
```

      Setting this parameter protects against bad workloads and identifies queries that need to be optimized. To resolve the failed queries:

      - Apply the appropriate partition filters.
      - Override the limit on a per-query basis.
      - Increase the cluster-wide limit beyond 1000, if needed, but note that this adds memory pressure to HiveServer2 and the Hive metastore.

    - If the accessed table is not partitioned, see this Cloudera Engineering Blog post, which explains how to partition Hive tables to improve query performance. Choose columns or dimensions for partitioning based upon usage patterns. Partitioning tables too much causes data fragmentation, but partitioning too little causes queries to read too much data. Either extreme makes querying inefficient. Typically, a few thousand table partitions is fine.

- **Wide tables or columns:**

  - Memory requirements are directly proportional to the number of columns and the size of the individual columns. Typically, a wide table contains over 1,000 columns. Wide tables or columns can cause memory pressure if the number of columns is large. This is especially true for Parquet files because all data for a row-group must be in memory before it can be written to disk. Avoid wide tables when possible.
  - Large individual columns also cause the memory requirements to increase. Typically, this happens when a column contains free-form text or complex types.

    How to resolve:

    - Reduce the total number of columns that are materialized. If only a subset of columns are required, avoid `SELECT *` because it materializes all columns.
    - Instead, use a specific set of columns. This is particularly efficient for wide tables that are stored in column formats. Specify columns explicitly instead of using `SELECT *`, especially for production workloads.

- **High query complexity**

  Complex queries usually have large numbers of joins, often over 10 joins per query. HS2 heap size requirements increase significantly as the number of joins in a query increases.

  How to resolve:

  - Make sure that partition filters are specified on all partitioned tables that are involved in JOINs.
  - Whenever possible, break queries into multiple smaller queries with intermediate temporary tables.

- **Improperly written user-defined functions (UDFs)**

Improperly written UDFs can exert significant memory pressure on HS2.

How to resolve:

– Understand the memory implications of the UDF and test it before using it in production environments.

General Best Practices

The following general best practices help maintain a healthy Hive cluster:

- Review and test queries in a development or test cluster before running them in a production environment. Monitor heap memory usage while testing.
- Redirect and isolate any untested, unreviewed, ad-hoc, or "dangerous" queries to a separate HS2 instance that is not critical to batch operation.

Optimizing Hive Write Performance on Amazon S3

In CDH 5.9 and later, support has been added to improve write performance to tables and partitions whose underlying files reside on the S3 filesystem (see Hive-14270). When the optimization is enabled, Hive writes intermediate data to HDFS when performing inserts to tables or partitions on S3, avoiding expensive S3 copy operations. The configuration is off by default when you upgrade, but can be enabled with the following configuration steps.

See Configuring Transient Hive ETL Jobs to Use the Amazon S3 Filesystem for information about setting up Hive to use S3.

Configuring Optimized Hive S3 Write Performance

To enable this optimization, you must set the following two configuration parameters:

- `hive.blobstore.use.as.scratchdir=false`

  This parameter changes the Hive scratch directory from S3 to HDFS.

- `hive.exec.copyfile.maxsize=1099511627776`

  This parameter disables `distcp` on files that are smaller than 1 TB.

> **Important:** Both configuration parameters must be set simultaneously to see performance gains. If the scratch directory setting is enabled alone without disabling `distcp`, performance degradation might occur.

Setting Parameters on a Per-Query Basis

Turn on this optimization feature on a per-query basis by setting these parameters in the query code with the Hive SET command:

```
set hive.blobstore.use.blobstore.as.scratchdir=false
set hive.exec.copyfile.maxsize=1099511627776;
```

Setting Parameters as Service-Wide Defaults with Cloudera Manager

Use Cloudera Manager to enable this feature by setting these same two parameters as service-wide defaults by using the HiveServer2 Safety Valve configuration setting for `hive-site.xml`:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. Click the **Configuration** tab.
3. Click the **HiveServer2** scope.
4. Click the **Advanced** category.

5. Scroll down to the **HiveServer2 Advanced Configuration Snippet (Safety Valve)** for `hive-site.xml` and click the plus sign under the last configuration parameter in this section.
6. In the new **Name** and **Value** fields, type the **hive.blobstore.use.blobstore.as.scratchdir** parameter name and the **false** value, respectively.
7. Click the plus sign again, and in the **Name** and **Value** fields, type the **hive.exec.copyfile.maxsize** parameter name and the **1099511627776** value.
8. Click **Save Changes**.

Restrictions

The following restrictions apply to this feature:

- **CTAS (CREATE TABLE AS SELECT) Queries**

  Setting these parameters might not improve performance when the following two conditions are true:

  - The target table's parent database is created with an HDFS location, and
  - The individual CTAS query specifies an S3 location.

  Performance gains do not occur if CTAS queries have already staged data in HDFS prior to installing CDH 5.9. However, if the target database of a CTAS query is created with a location that specifies an S3 path, query performance might be improved when CDH 5.9 is installed due to this optimization. Enabling the optimization in either case causes no harm.

- **HDFS Encryption**

  The `hive.exec.copyfile.maxsize` parameter is also used when copying files between encryption zones for `LOAD` statements. If you are using HDFS encryption in an environment where performance is critical, Cloudera does not recommend that you change the service-wide value of `hive.exec.copyfile.maxsize`.

Tuning Hive on Spark

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

Hive on Spark provides better performance than Hive on MapReduce while offering the same features. Running Hive on Spark requires no changes to user queries. Specifically, user-defined functions (UDFs) are fully supported, and most performance-related configurations work with the same semantics.

This topic describes how to configure and tune Hive on Spark for optimal performance. This topic assumes that your cluster is managed by Cloudera Manager and that you use YARN as the Spark cluster manager.

The example described in the following sections assumes a 40-host YARN cluster, and each host has 32 cores and 120 GB memory.

YARN Configuration

The YARN properties `yarn.nodemanager.resource.cpu-vcores` and `yarn.nodemanager.resource.memory-mb` determine how cluster resources can be used by Hive on Spark (and other YARN applications). The values for the two properties are determined by the capacity of your host and the number of other non-YARN applications that coexist on the same host. Most commonly, only YARN NodeManager and HDFS DataNode services are running on worker hosts.

Configuring Cores

Allocate 1 core for each of the services and 2 additional cores for OS usage, leaving 28 cores available for YARN.

Configuring Memory

Allocate 20 GB memory for these services and processes. To do so, set `yarn.nodemanager.resource.memory-mb=100 GB` and `yarn.nodemanager.resource.cpu-vcores=28`.

For more information on tuning YARN, see Tuning YARN.

Spark Configuration

After allocating resources to YARN, you define how Spark uses the resources: executor and driver memory, executor allocation, and parallelism.

Configuring Executor Memory

Spark executor configurations are described in Configuring Spark on YARN Applications.

When setting executor memory size, consider the following factors:

- More executor memory enables map join optimization for more queries, but can result in increased overhead due to garbage collection.
- In some cases the HDFS client does not handle concurrent writers well, so a race condition can occur if an executor has too many cores.

To minimize the number of unused cores, Cloudera recommends setting `spark.executor.cores` to 4, 5, or 6, depending on the number of cores allocated for YARN.

Because 28 cores is divisible by 4, set `spark.executor.cores` to 4. Setting it to 6 would leave 4 cores unused ; setting it to 5 leaves 3 cores unused. With `spark.executor.cores` set to 4, the maximum number of executors that can run concurrently on a host is seven (28 / 4). Divide the total memory among these executors, with each getting approximately 14 GB (100 / 7).

The total memory allocated to an executor includes `spark.executor.memory` and `spark.yarn.executor.memoryOverhead`. Cloudera recommends setting `spark.yarn.executor.memoryOverhead` to 15-20% of the total memory size that is, set `spark.executor.memoryOverhead=2 G` and `spark.executor.memory=12 G`.

With these configurations, each host can run up to 7 executors at a time. Each executor can run up to 4 tasks (one per core). So, each task has on average 3.5 GB (14 / 4) memory. All tasks running in an executor share the same heap space.

Make sure the sum of `spark.executor.memoryOverhead` and `spark.executor.memory` is less than `yarn.scheduler.maximum-allocation-mb`.

Configuring Driver Memory

You must also configure Spark driver memory:

- `spark.driver.memory`—Maximum size of each Spark driver's Java heap memory when Hive is running on Spark.
- `spark.yarn.driver.memoryOverhead`—Amount of extra off-heap memory that can be requested from YARN, per driver. This, together with `spark.driver.memory`, is the total memory that YARN can use to create a JVM for a driver process.

Spark driver memory does not impact performance directly, but it ensures that the Spark jobs run without memory constraints at the driver. Adjust the total amount of memory allocated to a Spark driver by using the following formula, assuming the value of `yarn.nodemanager.resource.memory-mb` is:

- 12 GB when *X* is greater than 50 GB
- 4 GB when *X* is between 12 GB and 50 GB
- 1 GB when *X* is between 1GB and 12 GB
- 256 MB when *X* is less than 1 GB

These numbers are for the sum of `spark.driver.memory` and `spark.yarn.driver.memoryOverhead`. Overhead should be 10-15% of the total. In this example, `yarn.nodemanager.resource.memory-mb=100 GB`, so the total memory for the Spark driver can be set to 12 GB. As a result, memory settings are `spark.driver.memory=10.5gb` and `spark.yarn.driver.memoryOverhead=1.5gb`.

Choosing the Number of Executors

The number of executors for a cluster is determined by the number of executors on each host and the number of worker hosts in the cluster. If you have 40 worker hosts in your cluster, the maximum number of executors that Hive can use to run Hive on Spark jobs is 160 (40 x 4). The maximum is slightly smaller than this because the driver uses one core and 12 GB total driver memory. This assumes that no other YARN applications are running.

Hive performance is directly related to the number of executors used to run a query. However, the characteristics vary from query to query. In general, performance is proportional to the number of executors. For example, using four executors for a query takes approximately half of the time of using two executors. However, performance peaks at a certain number of executors, above which increasing the number does not improve performance and can have an adverse impact.

In most cases, using half of the cluster capacity (half the number of executors) provides good performance. To achieve *maximum* performance, it is a good idea to use all available executors. For example, set `spark.executor.instances=160`. For benchmarking and performance measurement, this is strongly recommended.

Dynamic Executor Allocation

Although setting `spark.executor.instances` to the maximum value usually maximizes performance, doing so is not recommended for a production environment in which multiple users are running Hive queries. Avoid allocating a fixed number of executors for a user session, because the executors cannot be used by other user queries if they are idle. In a production environment, plan for executor allocation that allows greater resource sharing.

Spark allows you to dynamically scale the set of cluster resources allocated to a Spark application based on the workload. To enable dynamic allocation, follow the procedure in Dynamic Allocation. Except in certain circumstances, Cloudera strongly recommends enabling dynamic allocation.

Parallelism

For available executors to be fully utilized you must run enough tasks concurrently (in parallel). In most cases, Hive determines parallelism automatically for you, but you may have some control in tuning concurrency. On the input side, the number of map tasks is equal to the number of splits generated by the input format. For Hive on Spark, the input format is `CombineHiveInputFormat`, which can group the splits generated by the underlying input formats as required. You have more control over parallelism at the stage boundary. Adjust `hive.exec.reducers.bytes.per.reducer` to control how much data each reducer processes, and Hive determines an optimal number of partitions, based on the available executors, executor memory settings, the value you set for the property, and other factors. Experiments show that Spark is less sensitive than MapReduce to the value you specify for `hive.exec.reducers.bytes.per.reducer`, as long as enough tasks are generated to keep all available executors busy. For optimal performance, pick a value for the property so that Hive generates enough tasks to fully use all available executors.

For more information on tuning Spark applications, see Tuning Spark Applications.

Hive Configuration

Hive on Spark shares most if not all Hive performance-related configurations. You can tune those parameters much as you would for MapReduce. However, `hive.auto.convert.join.noconditionaltask.size`, which is the threshold for converting common join to map join based on statistics, can have a significant performance impact. Although this configuration is used for both Hive on MapReduce and Hive on Spark, it is interpreted differently by each.

The size of data is described by two statistics:

- `totalSize`—Approximate size of data on disk
- `rawDataSize`—Approximate size of data in memory

Hive on MapReduce uses `totalSize`. When both are available, Hive on Spark uses `rawDataSize`. Because of compression and serialization, a large difference between `totalSize` and `rawDataSize` can occur for the same dataset. For Hive on Spark, you might need to specify a larger value for `hive.auto.convert.join.noconditionaltask.size` to convert the same join to a map join. You can increase the value for this parameter to make map join conversion more aggressive. Converting common joins to map joins can improve performance. Alternatively, if this value is set too high, too much memory is used by data from small tables, and tasks may fail because they run out of memory. Adjust this value according to your cluster environment.

You can control whether `rawDataSize` statistics should be collected, using the property `hive.stats.collect.rawdatasize`. Cloudera recommends setting this to `true` in Hive (the default).

Cloudera also recommends setting two additional configuration properties, using a Cloudera Manager advanced configuration snippet for HiveServer2:

- `hive.stats.fetch.column.stats=true`
- `hive.optimize.index.filter=true`

The following properties are generally recommended for Hive performance tuning, although they are not specific to Hive on Spark:

```
hive.optimize.reducededuplication.min.reducer=4
hive.optimize.reducededuplication=true
hive.merge.mapfiles=true
hive.merge.mapredfiles=false
hive.merge.smallfiles.avgsize=16000000
hive.merge.size.per.task=256000000
hive.merge.sparkfiles=true
hive.auto.convert.join=true
hive.auto.convert.join.noconditionaltask=true
hive.auto.convert.join.noconditionaltask.size=20M(might need to increase for Spark,
200M)
hive.optimize.bucketmapjoin.sortedmerge=false
hive.map.aggr.hash.percentmemory=0.5
hive.map.aggr=true
hive.optimize.sort.dynamic.partition=false
hive.stats.autogather=true
hive.stats.fetch.column.stats=true
hive.compute.query.using.stats=true
hive.limit.pushdown.memory.usage=0.4 (MR and Spark)
hive.optimize.index.filter=true
hive.exec.reducers.bytes.per.reducer=67108864
hive.smbjoin.cache.rows=10000
hive.fetch.task.conversion=more
hive.fetch.task.conversion.threshold=1073741824
hive.optimize.ppd=true
```

Pre-warming YARN Containers

When you submit your first query after starting a new session, you may experience a slightly longer delay before you see the query start. You may also notice that if you run the same query again, it finishes much faster than the first one.

Spark executors need extra time to start and initialize for the Spark on YARN cluster, which causes longer latency. In addition, Spark does not wait for all executors to be ready before starting the job so some executors may be still starting up after the job is submitted to the cluster. However, for jobs running on Spark, the number of available executors at the time of job submission partly determines the number of reducers. When the number of ready executors has not reached the maximum, the job may not have maximal parallelism. This can further impact performance for the first job.

In long-lived user sessions, this extra time causes no problems because it only happens on the first query execution. However short-lived sessions, such as Hive jobs launched by Oozie, may not achieve optimal performance.

To reduce startup time, you can enable container pre-warming before a job starts. The job starts running only when the requested executors are ready. This way, a short-lived session parallelism is not decreased on the reduce side.

To enable pre-warming, set `hive.prewarm.enabled` to `true` before the query is issued. You can also set the umber of containers by setting `hive.prewarm.numcontainers`. The default is 10.

The actual number of executors to pre-warm is capped by the value of either `spark.executor.instances` (static allocation) or `spark.dynamicAllocation.maxExecutors` (dynamic allocation). The value for `hive.prewarm.numcontainers` should not exceed that allocated to a user session.

> **Note:** Pre-warming takes a few seconds and is a good practice for short-lived sessions, especially if the query involves reduce stages. However, if the value of `hive.prewarm.numcontainers` is higher than what is available in the cluster, the process can take a maximum of 30 seconds. Use pre-warming with caution.

Hive Metastore High Availability

You can enable Hive metastore high availability (HA) so that your cluster is resilient to failures if a metastore becomes unavailable.

Prerequisites

- Cloudera recommends that each instance of the metastore runs on a separate cluster host, to maximize high availability.
- Hive metastore HA requires a database that is also highly available, such as MySQL with replication in active-active mode. Refer to the documentation for your database of choice to configure it correctly.

Limitations

Sentry HDFS synchronization does not support Hive metastore HA.

Enabling Hive Metastore High Availability Using Cloudera Manager

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

1. Go to the Hive service.
2. If you have a secure cluster, change the Hive Delegation Token Store implementation. Non-secure clusters can skip this step.

   If more than one role group applies to this configuration, edit the value for the appropriate role group. See Modifying Configuration Properties Using Cloudera Manager.

   a. Click the **Configuration** tab.
   b. Select **Scope** > **Hive Metastore Server**.
   c. Select **Category** > **Advanced**.
   d. Locate the **Hive Metastore Delegation Token Store** property or search for it by typing its name In the Search box.
   e. Select `org.apache.hadoop.hive.thrift.DBTokenStore`.
   f. Click **Save Changes** to commit the changes.

3. Click the **Instances** tab.
4. Click **Add Role Instances**.
5. Click the text field under **Hive Metastore Server**.
6. Check the box by the host on which to run the additional metastore and click **OK**.
7. Click **Continue** and click **Finish**.
8. Check the box by the new **Hive Metastore Server** role.
9. Select **Actions for Selected** > **Start**, and click **Start** to confirm.
10. Click **Close** and click ⏻ to display the stale configurations page.
11. Click **Restart Stale Services** and click **Restart Now**.
12. Click **Finish** after the cluster finishes restarting.

Enabling Hive Metastore High Availability Using the Command Line

To configure the Hive metastore for high availability, configure each metastore to store its state in a replicated database, then provide the metastore clients with a list of URIs where metastores are available. The client starts with the first URI in the list. If it does not get a response, it randomly picks another URI in the list and attempts to connect. This continues until the client receives a response.

> **Important:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

1. Configure Hive on each of the cluster hosts where you want to run a metastore, following the instructions at Configuring the Hive Metastore on page 19.
2. On the server where the master metastore instance runs, edit the `/etc/hive/conf.server/hive-site.xml` file, setting the `hive.metastore.uris` property's value to a list of URIs where a Hive metastore is available for failover.

```
<property>
 <name>hive.metastore.uris</name>

<value>thrift://metastore1.example.com,thrift://metastore2.example.com,thrift://metastore3.example.com</value>

 <description> URI for client to contact metastore server </description>
</property>
```

3. If you use a secure cluster, enable the Hive token store by configuring the value of the `hive.cluster.delegation.token.store.class` property to `org.apache.hadoop.hive.thrift.DBTokenStore`. Non-secure clusters can skip this step.

```
<property>
 <name>hive.cluster.delegation.token.store.class</name>
 <value>org.apache.hadoop.hive.thrift.DBTokenStore</value>
</property>
```

4. Save your changes and restart each Hive instance.
5. Connect to each metastore and update it to use a nameservice instead of a NameNode, as a requirement for high availability.

   a. From the command-line, as the Hive user, retrieve the list of URIs representing the filesystem roots:

   ```
   hive --service metatool -listFSRoot
   ```

   b. Run the following command with the `--dry-run` option, to be sure that the nameservice is available and configured correctly. This will not change your configuration.

   ```
   hive --service metatool -updateLocation nameservice-uri namenode-uri -dryRun
   ```

   c. Run the same command again without the `--dry-run` option to direct the metastore to use the nameservice instead of a NameNode.

   ```
   hive --service metatool -updateLocation nameservice-uri namenode-uri
   ```

6. Test your configuration by stopping your main metastore instance, and then attempting to connect to one of the other metastores from a client. The following is an example of doing this on a RHEL or Fedora system. The example first stops the local metastore, then connects to the metastore on the host metastore2.example.com and runs the `SHOW TABLES` command.

```
$ sudo service hive-metastore stop
$ /usr/lib/hive/bin/beeline
beeline> !connect jdbc:hive2://metastore2.example.com:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000> SHOW TABLES;
show tables;
+-----------+
| tab_name  |
+-----------+
+-----------+
```

```
No rows selected (0.238 seconds)
0: jdbc:hive2://localhost:10000>
```

7. Restart the local metastore when you have finished testing.

```
$ sudo service hive-metastore start
```

Configuring HiveServer2 High Availability in CDH

To enable high availability for multiple HiveServer2 hosts, configure a load balancer to manage them. To increase stability and security, configure the load balancer on a proxy server. The following sections describe how to enable high availability by using Cloudera Manager or how to enable it manually for unmanaged clusters.

Warning:

- HiveServer2 high availability does not automatically fail and retry long-running Hive queries. If any of the HiveServer2 instances fail, all queries running on that instance fail and are not retried. Instead, the client application must re-submit the queries.
- After you enable HiveServer2 high availability, existing Oozie jobs must be changed to reflect the HiveServer2 address.
- On Kerberos-enabled clusters, you *must* use the load balancer for all connections. After you enable HiveServer2 high availability, direct connections to HiveServer2 instances fail.

Enabling HiveServer2 High Availability Using Cloudera Manager

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

1. Add multiple HiveServer2 instances to your cluster:
    1. Go to the **Hive** service.
    2. Click the **Instances** tab, and then click **Add Role Instances**.
    3. On the **Add Role Instances to Hive** page under the **HiveServer2** column heading, click **Select hosts**, and select the hosts that should have a HiveServer2 instance.
    4. Click **OK**, and then click **Continue**. You will be brought to the Instances page where you can start the new HiveServer2 instances.
2. Click the **Configuration** tab.
3. Select **Scope** > **HiveServer2**.
4. Select **Category** > **Main**.
5. Locate the **HiveServer2 Load Balancer** property or search for it by typing its name in the Search box.
6. Enter values for `<hostname>:<port number>`. For example, `hs2load_balancer.example.com:10015`.

    Note: When you set the **HiveServer2 Load Balancer** property, Cloudera Manager regenerates the keytabs for HiveServer2 roles. The principal in these keytabs contains the load balancer hostname. If there is a Hue service that depends on this Hive service, it also uses the load balancer to communicate with Hive.

7. Click **Save Changes** to commit the changes.
8. **Restart** the Hive service.

Configuring HiveServer2 to Load Balance Behind a Proxy on Unmanaged Clusters

For unmanaged clusters with multiple users and availability requirements, you can configure a proxy server to relay requests to and from each HiveServer2 host. Applications connect to a single well-known host and port, and connection requests to the proxy succeed even when hosts running HiveServer2 become unavailable.

Unmanaged Clusters with Kerberos Enabled

1. Create the `hive/<load_balancer_fully_qualified_domain_name>` principal and the merged keytab.

    - **If you are using MIT Kerberos**, connect to the KDC as `root` and run the following command in `kadmin.local`. Replace `<load_balancer_fully_qualified_domain_name>` with the fully qualified domain name of the load balancer host:

```
kadmin.local:  addprinc -randkey hive/<load_balancer_fully_qualified_domain_name>
```

    - **If you are using Microsoft Active Directory for your KDC**, see Microsoft documentation to create a principal and keytab for Hive. The principal must be named `hive/<load_balancer_fully_qualified_domain_name>` and the keytab must contain all of the Hive host keytabs for your cluster.

      For example, if your load balancer is `hs2loadbalancer.example.com` and you have two HiveServer2 instances on host `hs2-host-1.example.com` and `hs2-host-2.example.com`, if you run `klist -ekt hive-proxy.keytab`, it should return the following:

```
[root@cdh_user-linux named]# klist -ekt /tmp/hive-proxy.keytab
Keytab name: FILE:/tmp/hive-proxy.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------------
   1 09/08/2015 12:46:25 hive/hs2loadbalancer.example.com@EXAMPLE.COM
(aes256-cts-hmac-sha1-96)
   2 09/08/2015 12:46:37 hive/hs2-host-1.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)

   2 09/08/2015 12:46:42 hive/hs2-host-2.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
```

2. While you are still connected to `kadmin.local`, list the `hive/<hs2_hostname>` principals:

```
kadmin.local:  listprincs hive/*
hive/hs2-host-1.example.com@EXAMPLE.COM
hive/hs2-host-2.example.com@EXAMPLE.COM
```

3. While you are still connected to `kadmin.local`, create a `hive-proxy.keytab`, which contains the load balancer and all of the `hive/<hs2_hostname>` principals:

```
kadmin.local:  xst -k /tmp/hive-proxy.keytab -norandkey hive/hs2loadbalancer.example.com
kadmin.local:  xst -k /tmp/hive-proxy.keytab -norandkey hive/hs2-host-1.example.com
kadmin.local:  xst -k /tmp/hive-proxy.keytab -norandkey hive/hs2-host-2.example.com
```

   Note that a single `xst` is used per entry, which appends each entry to the keytab. Also note that the `-norandkey` parameter is specified. This is required so you do not break existing keytabs.

4. Validate the keytab by running `klist`:

```
[root@cdh_user-linux named]# klist -ekt /tmp/hive-proxy.keytab
Keytab name: FILE:/tmp/hive-proxy.keytab
KVNO Timestamp           Principal
---- ------------------- ------------------------------------------------------
   1 09/08/2015 12:46:25 hive/hs2loadbalancer.example.com@EXAMPLE.COM
(aes256-cts-hmac-sha1-96)
   2 09/08/2015 12:46:37 hive/hs2-host-1.example.com@EXAMPLE.COM (aes256-cts-hmac-sha1-96)
   2 09/08/2015 12:46:42 hive/hs2-host-2.example.com@EXAMPLE.COM (aes256-cts-hmca-sha1-96)
```

5. Distribute the `hive-proxy.keytab` to all HiveServer2 hosts. Make sure that `/var/lib/hive` exists on each node and copy the `hive-proxy.keytab` to `/var/lib/hive` on each node. Then confirm that permissions are set to `hive:hive` on the directory and the keytab:

```
[root@cdh5xx-1 ~]# rm -f /var/lib/hive/hive-proxy.keytab
[root@cdh5xx-1 ~]# mkdir -p /var/lib/hive
[root@cdh5xx-1 ~]# cp /tmp/hive-proxy.keytab /var/lib/hive
[root@cdh5xx-1 ~]# chown -R hive:hive /var/lib/hive
[root@cdh5xx-1 ~]# ls -lart /var/lib/hive/
total 16
drwxr-xr-x. 49 root root 4096 Jun  7 17:39 ..
-rw-r--r--   1 hive hive  983 Jun  7 17:40 hive.keystore
-rw-------   1 hive hive 1412 Sep  8 14:38 hive-proxy.keytab
drwxr-xr-x   2 hive hive 4096 Sep  8 14:38 .
```

6. Configure HiveServer2 to use the new keytab and load balancer principal by setting the `hive.server2.authentication.kerberos.principal` and the `hive.server2.authentication.kerberos.keytab` properties in the `hive-site.xml` file. For example, to set these properties for the examples used in the above steps, your `hive-site.xml` is set as follows:

```
<property>
   <name>hive.server2.authentication.kerberos.principal</name>
   <value>hive/hs2loadbalancer.example.com@EXAMPLE.COM</value>
</property>
<property>
   <name>hive.server2.authentication.kerberos.keytab</name>
   <value>/var/lib/hive/hive-proxy.keytab</value>
</property>
```

7. Restart the Hive service.
8. Download load-balancing proxy software of your choice on a single host. For example, see Example HAProxy Configuration.
9. Configure the software, typically by editing a configuration file. Usually this configuration includes:

   a. Setting the port for the load balancer to listen on and to relay HiveServer2 requests back and forth.

   b. Setting the port and hostname for each HiveServer2 host. These are the hosts from which the load balancer chooses when relaying each query.

10. Run the load-balancing proxy server and point it at the configuration file.
11. Point all scripts, jobs, or application configurations to the new proxy server instead of any specific HiveServer2 instance.

Unmanaged Clusters WITHOUT Kerberos

To configure HiveServer2 for high availability for unmanaged clusters, use the following steps.

1. Download load-balancing proxy software of your choice on a single host. For example, see Example HAProxy Configuration.
2. Configure the software, typically by editing a configuration file. Usually this configuration includes:

   a. Setting the port for the load balancer to listen on and to relay HiveServer2 requests back and forth.

   b. Setting the port and hostname for each HiveServer2 host. These are the hosts from which the load balancer chooses when relaying each query.

3. Run the load-balancing proxy server and point it at the configuration file.

4. Point all scripts, jobs, or application configurations to the new proxy server instead of any specific HiveServer2 instance.

Example HAProxy Configuration

To install and configure HAProxy, an open source load balancer, perform the following steps. For more information about HAProxy, see the HAProxy Community Edition web site.

1. Download the appropriate HAProxy software.
2. As the `root` user, install HAProxy:

```
sudo yum -y install haproxy
```

3. Edit the HAProxy configuration file to listen on port `10000` and point to each HiveServer2 instance. Make sure to configure for sticky sessions. Here is an example configuration file:

```
global
    # To have these messages end up in /var/log/haproxy.log you will
    # need to:
    #
    # 1) configure syslog to accept network log events.  This is done
    #    by adding the '-r' option to the SYSLOGD_OPTIONS in
    #    /etc/sysconfig/syslog
    #
    # 2) configure local2 events to go to the /var/log/haproxy.log
    #   file. A line like the following can be added to
    #   /etc/sysconfig/syslog
    #
    #    local2.*                       /var/log/haproxy.log
    #
    log         127.0.0.1 local0
    log         127.0.0.1 local1 notice
    chroot      /var/lib/haproxy
    pidfile     /var/run/haproxy.pid
    maxconn     4000
    user        haproxy
    group       haproxy
    daemon

    # turn on stats unix socket
    #stats socket /var/lib/haproxy/stats

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#
# You might need to adjust timing values to prevent timeouts.
#---------------------------------------------------------------------
defaults
    mode                    http
    log                     global
    option                  httplog
    option                  dontlognull
    option http-server-close
    option forwardfor       except 127.0.0.0/8
    option                  redispatch
    retries                 3
    maxconn                 3000
    contimeout 5000
    clitimeout 50000
    srvtimeout 50000

#
# This sets up the admin page for HA Proxy at port 25002.
#
listen stats :25002
    balance
    mode http
    stats enable
    stats auth username:password

# This is the setup for HS2. beeline client connect to load_balancer_host:10001.
# HAProxy will balance connections among the list of servers listed below.
listen hiveserver2 :10001
    mode tcp
    option tcplog
    balance source

    server hiveserver2_1 hs2-host-1.example.com:10000
    server hiveserver2_2 hs2-host-2.example.com:10000
    server hiveserver2_3 hs2-host-3.example.com:10000
    server hiveserver2_4 hs2-host-4.example.com:10000
```

4. Set HAProxy to start when the system starts:

```
chkconfig haproxy on
```

5. Start HAProxy:

```
service haproxy start
```

Hive Replication

Minimum Required Role: BDR Administrator (also provided by **Full Administrator**)

Hive replication enables you to copy (replicate) your Hive metastore and data from one cluster to another and synchronize the Hive metastore and data set on the *destination* cluster with the source, based on a specified replication schedule. The destination cluster must be managed by the Cloudera Manager Server where the replication is being set up, and the *source* cluster can be managed by that same server or by a peer Cloudera Manager Server.

Configuration notes:

- If the `hadoop.proxyuser.hive.groups` configuration has been changed to restrict access to the Hive Metastore Server to certain users or groups, the `hdfs` group or a group containing the `hdfs` user must also be included in the list of groups specified for Hive replication to work. This configuration can be specified either on the Hive service as an override, or in the core-site HDFS configuration. This applies to configuration settings on both the source and destination clusters.
- If you configured Synchronizing HDFS ACLs and Sentry Permissions on the target cluster for the directory where HDFS data is copied during Hive replication, the permissions that were copied during replication, are overwritten by the HDFS ACL synchronization and are not preserved
- If you are using Kerberos to secure your clusters, see Enabling Replication Between Clusters in Different Kerberos Realms for details about configuring it.

Network Latency and Replication

High latency among clusters can cause replication jobs to run more slowly, but does not cause them to fail. For best performance, latency between the source cluster NameNode and the destination cluster NameNode should be less than 80 milliseconds. (You can test latency using the Linux `ping` command.) Cloudera has successfully tested replications with latency of up to 360 milliseconds. As latency increases, replication performance degrades.

Host Selection for Hive Replication

If your cluster has Hive clients installed on hosts with limited resources, Hive replication may use these hosts to run commands for the replication, which can cause the performance of the replication to degrade. To improve performance, you can specify the hosts (a ”white list”) to use during replication so that the lower-resource hosts are not used.

To configure the hosts used for Hive Replication:

1. Click **Clusters** > **Hive** > **Configuration**.
2. Type `Hive Replication` in the search box.
3. Locate the **Hive Replication Environment Advanced Configuration Snippet (Safety Valve)** property.
4. Add the `HOST_WHITELIST` property. Enter a comma-separated list of hostnames to use for Hive replication. For example:

```
HOST_WHITELIST=host-1.mycompany.com,host-2.mycompany.com
```

5. Click **Save Changes** to commit the changes.

Hive Tables and DDL Commands

The following applies when using the `drop table` and `truncate table` DDL commands:

- If you configure replication of a Hive table and then later drop that table, the table remains on the destination cluster. The table is not dropped when subsequent replications occur.

- If you drop a table on the destination cluster, and the table is still included in the replication job, the table is re-created on the destination during the replication.
- If you drop a table partition or index on the source cluster, the replication job also drops them on the destination cluster.
- If you truncate a table, and the **Delete Policy** for the replication job is set to **Delete to Trash** or **Delete Permanently**, the corresponding data files are deleted on the destination during a replication.

Performance and Scalability Limitations

Hive replication has the following limitations:

- Maximum number of databases: 100
- Maximum number of tables per database: 1,000
- Maximum number of partitions per table: 10,000. See Identify Workload Characteristics That Increase Memory Pressure on page 60.
- Maximum total number of tables (across all databases): 10,000
- Maximum total number of partitions (across all tables): 100,000
- Maximum number of indexes per table: 100

Hive Replication in Dynamic Environments

To use BDR for Hive replication in environments where the Hive Metastore changes, such as when a database or table gets created or deleted, additional configuration is needed.

1. Open the Cloudera Manager Admin Console.
2. Search for the **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** property on the source cluster.
3. Add the following properties:

   - **Name**: `replication.hive.ignoreDatabaseNotFound`

     Value: `true`

   - **Name**:`replication.hive.ignoreTableNotFound`

     Value: `true`

4. Save the changes.
5. Restart the HDFS service.

Configuring Replication of Hive Data

1. Verify that your cluster conforms to one of the Supported Replication Scenarios.
2. If the source cluster is managed by a different Cloudera Manager server than the destination cluster, configure a peer relationship. If the source or destination is Amazon S3, you must configure an External Account.
3. Do one of the following:

   - From the **Backup** tab, select **Replications**.
   - From the **Clusters** tab, go to the Hive service and select **Quick Links** > **Replication**.

   The Schedules tab of the Replications page displays.

4. Select **Create New Schedule** > **Hive Replication**. The **General** tab displays.
5. Select the **General** tab to configure the following:

> **Note:** If you are replication to or from Amazon S3, follow the steps under Hive Replication To and From Amazon S3 on page 85 before completing these steps.

a. Use the **Source** drop-down list to select the cluster with the Hive service you want to replicate.

b. Use the **Destination** drop-down list to select the destination for the replication. If there is only one Hive service managed by Cloudera Manager available as a destination, this is specified as the destination. If more than one Hive service is managed by this Cloudera Manager, select from among them.

c. Leave **Replicate All** checked to replicate all the Hive databases from the source. To replicate only selected databases, uncheck this option and enter the database name(s) and tables you want to replicate.

- You can specify multiple databases and tables using the plus symbol to add more rows to the specification.
- You can specify multiple databases on a single line by separating their names with the pipe (|) character. For example: `mydbname1|mydbname2|mydbname3`.
- Regular expressions can be used in either database or table fields, as described in the following table:

Regular Expression	Result
`[\w].+`	Any database or table name.
`(?!myname\b).+`	Any database or table except the one named myname.

`db1|db2`
`[\w_]+`
	All tables of the db1 and db2 databases.

`db1`
`[\w_]+`
Click the "+" button and then enter
`db2`
`[\w_]+`
	All tables of the db1 and db2 databases (alternate method).

d. Select a **Schedule**:

- **Immediate** - Run the schedule Immediately.
- **Once** - Run the schedule one time in the future. Set the date and time.
- **Recurring** - Run the schedule periodically in the future. Set the date, time, and interval between runs.

e. To specify the user that should run the MapReduce job, use the **Run As Username** option. By default, MapReduce jobs run as `hdfs`. To run the MapReduce job as a different user, enter the user name. If you are using Kerberos, you *must* provide a user name here, and it must have an ID greater than 1000.

> **Note:** The user running the MapReduce job should have `read` and `execute` permissions on the Hive warehouse directory on the *source* cluster. If you configure the replication job to preserve permissions, superuser privileges are required on the *destination* cluster.

f. (Optional) Enter the name of a resource pool in the **Scheduler Pool** field. The value you enter is used by the **MapReduce Service** you specified when Cloudera Manager executes the MapReduce job for the replication. The job specifies the value using one of these properties:

- MapReduce - Fair scheduler: `mapred.fairscheduler.pool`
- MapReduce - Capacity scheduler: `queue.name`
- YARN - `mapreduce.job.queuename`

6. Select the **Resources** tab to configure the following:

- **Maximum Map Slots** and **Maximum Bandwidth**- Limits for the number of map slots and for bandwidth per mapper. The default is 100 MB.

- **Replication Strategy** - Whether file replication should be static (the default) or dynamic. Static replication distributes file replication tasks among the mappers up front to achieve a uniform distribution based on file sizes. Dynamic replication distributes file replication tasks in small sets to the mappers, and as each mapper processes its tasks, it dynamically acquires and processes the next unallocated set of tasks.

7. Select the **Advanced** tab to specify an export location, modify the parameters of the MapReduce job that will perform the replication, and set other options. You can select a MapReduce service (if there is more than one in your cluster) and change the following parameters:

   - Uncheck the **Replicate HDFS Files** checkbox to skip replicating the associated data files.
   - If both the source and destination clusters use CDH 5.7.0 or later, uncheck the **Replicate Impala Metadata** checkbox to avoid redundant replication of Impala metadata. (This option only displays when supported by both source and destination clusters.)

     To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.7 or lower, see Impala Metadata Replication for information on when to select this option.

   - The **Force Overwrite** option, if checked, forces overwriting data in the destination metastore if incompatible changes are detected. For example, if the destination metastore was modified, and a new partition was added to a table, this option forces deletion of that partition, overwriting the table with the version found on the source.

     > **Important:** If the **Force Overwrite** option is not set, and the Hive replication process detects incompatible changes on the source cluster, Hive replication fails. This sometimes occurs with recurring replications, where the metadata associated with an existing database or table on the source cluster changes over time.

   - By default, Hive metadata is exported to a default HDFS location (`/user/${user.name}/.cm/hive`) and then imported from this HDFS file to the destination Hive metastore. In this example, `user.name` is the process user of the HDFS service on the *destination* cluster. To override the default HDFS location for this export file, specify a path in the **Export Path** field.

     > **Note:** In a Kerberized cluster, the HDFS principal on the *source* cluster must have `read`, `write`, and `execute` access to the **Export Path** directory on the *destination* cluster.

   - By default, Hive HDFS data files (for example, `/user/hive/warehouse/db1/t1`) are replicated to a location relative to "`/`" (in this example, to `/user/hive/warehouse/db1/t1`). To override the default, enter a path in the **HDFS Destination Path** field. For example, if you enter `/ReplicatedData`, the data files would be replicated to `/ReplicatedData/user/hive/warehouse/db1/t1`.
   - Select the **MapReduce Service** to use for this replication (if there is more than one in your cluster).
   - **Log Path** - An alternative path for the logs.
   - **Abort on Error** - Whether to abort the job on an error. By selecting the check box, files copied up to that point remain on the destination, but no additional files will be copied. Abort on Error is off by default.
   - **Skip Checksum Checks** - Whether to skip checksum checks, which are performed by default.

     Checksums are used for two purposes:

     - To skip replication of files that have already been copied. If **Skip Checksum Checks** is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination.
     - To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data.

   - **Delete Policy** - Whether files that were on the source should also be deleted from the destination directory. Options include:

  - **Keep Deleted Files** - Retains the destination files even when they no longer exist at the source. (This is the default.).
  - **Delete to Trash** - If the HDFS trash is enabled, files are moved to the trash folder. (Not supported when replicating to Amazon S3.)
  - **Delete Permanently** - Uses the least amount of space; use with caution.

- **Preserve** - Whether to preserve the **Block Size**, **Replication Count**, and **Permissions** as they exist on the source file system, or to use the settings as configured on the destination file system. By default, settings are preserved on the source.

> **Note:** You must be running as a superuser to preserve permissions. Use the "Run As Username" option to ensure that is the case.

- **Alerts** - Whether to generate alerts for various state changes in the replication workflow. You can alert **On Failure**, **On Start**, **On Success**, or **On Abort** (when the replication workflow is aborted).

8. Click **Save Schedule**.

   The replication task appears as a row in the **Replications Schedule** table. See Viewing Replication Schedules on page 81.

To specify additional replication tasks, select **Create** > **Hive Replication**.

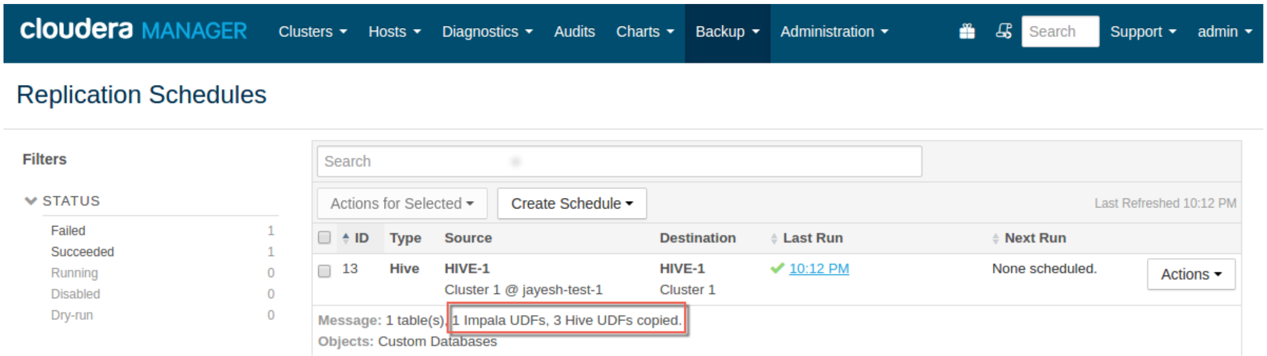
> **Note:** If your replication job takes a long time to complete, and tables change before the replication finishes, the replication may fail. Consider making the **Hive Warehouse Directory** and the directories of any external tables snapshottable, so that the replication job creates snapshots of the directories before copying the files. See Using Snapshots with Replication.

Replication of Impala and Hive User Defined Functions (UDFs)

By default, for clusters where the version of CDH is 5.7 or higher, Impala and Hive UDFs are persisted in the Hive Metastore and are replicated automatically as part of Hive replications. See Impala User-Defined Functions (UDFs), Impala Metadata Replication, and Managing User-Defined Functions (UDFs) with HiveServer2 on page 46.

To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.7 or lower, see Impala Metadata Replication for information on when to select the **Replicate Impala Metadata** option on the **Advanced** tab when creating a Hive replication schedule. .

After a replication has run, you can see the number of Impala and Hive UDFs that were replicated during the last run of the schedule on the **Replication Schedules** page:



Replication Schedules

STATUS

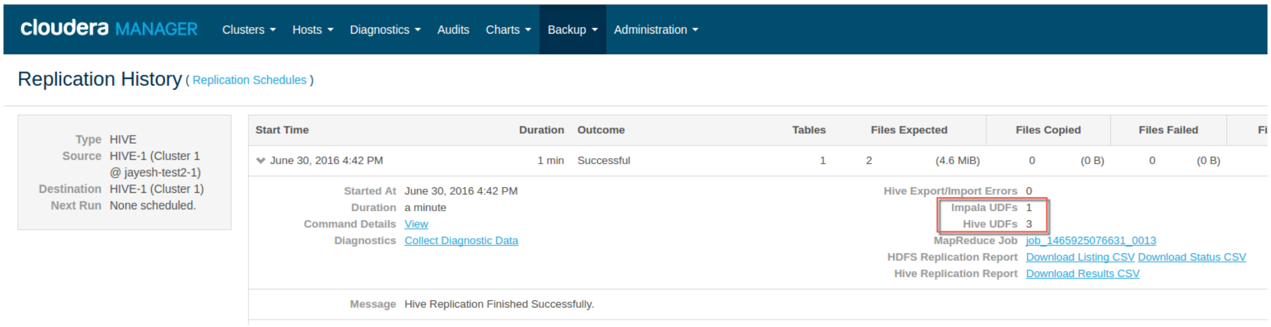
Failed	1
Succeeded	1
Running	0
Disabled	0
Dry-run	0

Last Refreshed 10:12 PM

ID	Type	Source	Destination	Last Run	Next Run
13	Hive	HIVE-1 Cluster 1 @ jayesh-test-1	HIVE-1 Cluster 1	✓ 10:12 PM	None scheduled.

Message: 1 table(s). 1 Impala UDFs, 3 Hive UDFs copied.
Objects: Custom Databases

For previously-run replications, the number of replicated UDFs displays on the **Replication History** page:



Replication History (Replication Schedules)

Type HIVE
Source HIVE-1 (Cluster 1 @ jayesh-test2-1)
Destination HIVE-1 (Cluster 1)
Next Run None scheduled.

Start Time	Duration	Outcome	Tables	Files Expected		Files Copied		Files Failed		Fi
June 30, 2016 4:42 PM	1 min	Successful	1	2	(4.6 MiB)	0	(0 B)	0	(0 B)	

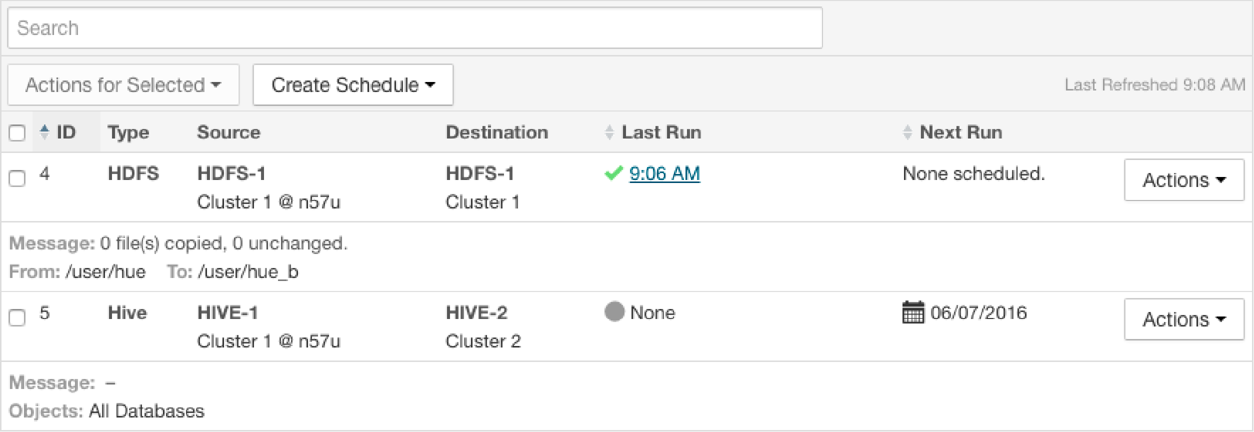
Started At June 30, 2016 4:42 PM
Duration a minute
Command Details View
Diagnostics Collect Diagnostic Data

Hive Export/Import Errors 0
Impala UDFs 1
Hive UDFs 3
MapReduce Job job_1465925076631_0013
HDFS Replication Report Download Listing CSV Download Status CSV
Hive Replication Report Download Results CSV

Message Hive Replication Finished Successfully.

Viewing Replication Schedules

The **Replications Schedules** page displays a row of information about each scheduled replication job. Each row also displays recent messages regarding the last time the Replication job ran.



Last Refreshed 9:08 AM

□	ID	Type	Source	Destination	Last Run	Next Run	
□	4	HDFS	HDFS-1 Cluster 1 @ n57u	HDFS-1 Cluster 1	✓ 9:06 AM	None scheduled.	Actions
□	5	Hive	HIVE-1 Cluster 1 @ n57u	HIVE-2 Cluster 2	● None	06/07/2016	Actions

Message: 0 file(s) copied, 0 unchanged.
From: /user/hue To: /user/hue_b

Message: –
Objects: All Databases

Figure 5: Replication Schedules Table

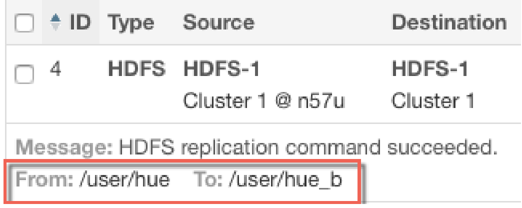
Only one job corresponding to a replication schedule can occur at a time; if another job associated with that same replication schedule starts before the previous one has finished, the second one is canceled.

You can limit the replication jobs that are displayed by selecting filters on the left. If you do not see an expected schedule, adjust or clear the filters. Use the search box to search the list of schedules for path, database, or table names.

The **Replication Schedules** columns are described in the following table.

Table 1: Replication Schedules Table

Column	Description
ID	An internally generated ID number that identifies the schedule. Provides a convenient way to identify a schedule. Click the **ID** column label to sort the replication schedule table by ID.
Type	The type of replication scheduled, either HDFS or Hive.
Source	The source cluster for the replication.
Destination	The destination cluster for the replication.
Objects	Displays on the bottom line of each row, depending on the type of replication: • **Hive** - A list of tables selected for replication.

Column	Description
	• **HDFS** - A list of paths selected for replication. For example:  □ ID Type Source Destination □ 4 HDFS HDFS-1 Cluster 1 @ n57u HDFS-1 Cluster 1 Message: HDFS replication command succeeded. From: /user/hue To: /user/hue_b
Last Run	The date and time when the replication last ran. Displays **None** if the scheduled replication has not yet been run. Click the date and time link to view the Replication History page for the replication. Displays one of the following icons: • ✓ - Successful. Displays the date and time of the last run replication. • ⊗ - Failed. Displays the date and time of a failed replication. • ● - None. This scheduled replication has not yet run. • - Running. Displays a spinner and bar showing the progress of the replication. Click the **Last Run** column label to sort the Replication Schedules table by the last run date.
Next Run	The date and time when the next replication is scheduled, based on the schedule parameters specified for the schedule. Hover over the date to view additional details about the scheduled replication. Click the **Next Run** column label to sort the Replication Schedules table by the next run date.
Actions	The following items are available from the **Action** button: • **Show History** - Opens the **Replication History** page for a replication. See Viewing Replication History. • **Edit Configuration** - Opens the **Edit Replication Schedule** page. • **Dry Run** - Simulates a run of the replication task but does not actually copy any files or tables. After a Dry Run, you can select **Show History**, which opens the **Replication History** page where you can view any error messages and the number and size of files or tables that would be copied in an actual replication. • Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for the last 10 runs of the schedule: 1. Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. 2. Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. 3. When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • **Run Now** - Runs the replication task immediately. • **Disable | Enable** - Disables or enables the replication schedule. No further replications are scheduled for disabled replication schedules. • **Delete** - Deletes the schedule. Deleting a replication schedule does not delete copied files or tables.

- While a job is in progress, the **Last Run** column displays a spinner and progress bar, and each stage of the replication task is indicated in the message beneath the job's row. Click the **Command Details** link to view details about the execution of the command.
- If the job is successful, the number of files copied is indicated. If there have been no changes to a file at the source since the previous job, then that file is *not* copied. As a result, after the initial job, only a subset of the files may actually be copied, and this is indicated in the success message.
- If the job fails, the ✖ icon displays.
- To view more information about a completed job, select **Actions** > **Show History**. See Viewing Replication History.

Enabling, Disabling, or Deleting A Replication Schedule

When you create a new replication schedule, it is automatically enabled. If you disable a replication schedule, it can be re-enabled at a later time.

To enable, disable, or delete a replication schedule, do one of the following:

- 1. Click **Actions** > **Enable|Disable|Delete** in the row for a replication schedule.

-or-

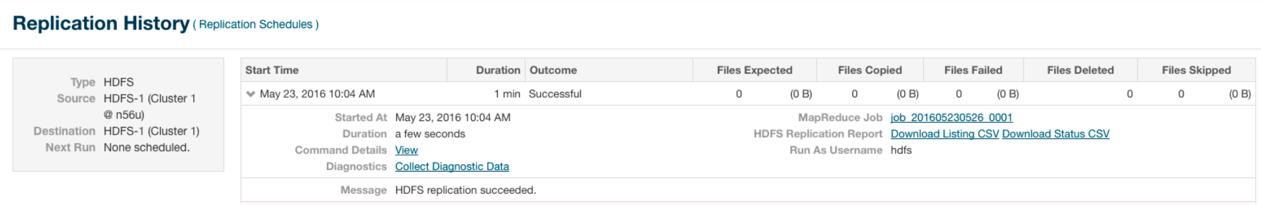
- 1. Select one or more replication schedules in the table by clicking the check box the in the left column of the table.
  2. Click **Actions for Selected** > **Enable|Disable|Delete**.

Viewing Replication History

You can view historical details about replication jobs on the **Replication History** page.

To view the history of a replication job:

1. Select **Backup** > **Replication Schedules** to go to the **Replication Schedules** page.
2. Locate the row for the job.
3. Click **Actions** > **Show History**.



Replication History (Replication Schedules)

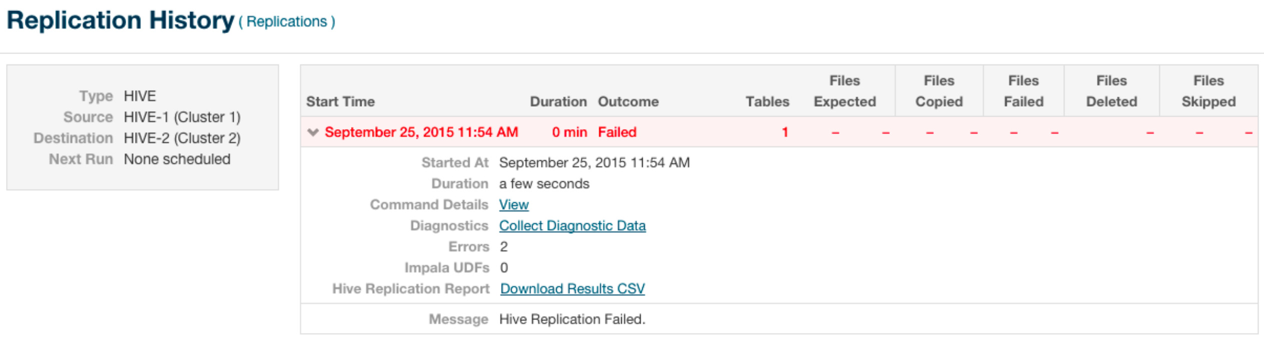
Type HDFS
Source HDFS-1 (Cluster 1 @ n56u)
Destination HDFS-1 (Cluster 1)
Next Run None scheduled.

Start Time	Duration	Outcome	Files Expected		Files Copied		Files Failed		Files Deleted	Files Skipped	
May 23, 2016 10:04 AM	1 min	Successful	0	(0 B)	0	(0 B)	0	(0 B)	0	0	(0 B)

Started At May 23, 2016 10:04 AM
Duration a few seconds
Command Details View
Diagnostics Collect Diagnostic Data
MapReduce Job job_201605230526_0001
HDFS Replication Report Download Listing CSV Download Status CSV
Run As Username hdfs

Message HDFS replication succeeded.

Figure 6: Replication History Screen (HDFS)



Replication History (Replications)

Type HIVE
Source HIVE-1 (Cluster 1)
Destination HIVE-2 (Cluster 2)
Next Run None scheduled

Start Time	Duration	Outcome	Tables	Files Expected		Files Copied		Files Failed		Files Deleted	Files Skipped	
September 25, 2015 11:54 AM	0 min	Failed	1	–	–	–	–	–	–	–	–	–

Started At September 25, 2015 11:54 AM
Duration a few seconds
Command Details View
Diagnostics Collect Diagnostic Data
Errors 2
Impala UDFs 0
Hive Replication Report Download Results CSV

Message Hive Replication Failed.

Figure 7: Replication History Screen (Hive, Failed Replication)

The **Replication History** page displays a table of previously run replication jobs with the following columns:

Table 2: Replication History Table

Column	Description
Start Time	Time when the replication job started. Click › to expand the display and show details of the replication. In this screen, you can: • Click the **View** link to open the **Command Details** page, which displays details and messages about each step in the execution of the command. Click › to expand the display for a **Step** to: – View the actual command string. – View the Start time and duration of the command. – Click the **Context** link to view the service status page relevant to the command. – Select one of the tabs to view the **Role Log**, **stdout**, and **stderr** for the command. See [Viewing Running and Recent Commands](). • Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for this run of the schedule: **1.** Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. **2.** Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. **3.** When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • (HDFS only) Link to view details on the **MapReduce Job** used for the replication. See [Viewing and Filtering MapReduce Activities](). • (Dry Run only) View the number of **Replicable Files**. Displays the number of files that would be replicated during an actual replication. • (Dry Run only) View the number of **Replicable Bytes**. Displays the number of bytes that would be replicated during an actual replication. • Link to download a CSV file containing a **Replication Report**. This file lists the databases and tables that were replicated. • View the number of **Errors** that occurred during the replication. • View the number of **Impala UDFs** replicated. (Displays only for Hive replications where **Replicate Impala Metadata** is selected.) • Click the link to download a CSV file containing a **Download Listing**. This file lists the files and directories that were replicated. • Click the link to download a CSV file containing **Download Status**. • If a user was specified in the **Run As Username** field when creating the replication job, the selected user displays. • View messages returned from the replication job.
Duration	Amount of time the replication job took to complete.
Outcome	Indicates success or failure of the replication job.
Files Expected	Number of files expected to be copied, based on the parameters of the replication schedule.
Files Copied	Number of files actually copied during the replication.
Tables	(Hive only) Number of tables replicated.
Files Failed	Number of files that failed to be copied during the replication.
Files Deleted	Number of files that were deleted during the replication.

Column	Description
Files Skipped	Number of files skipped during the replication. The replication process skips files that already exist in the destination and have not changed.

Hive Replication To and From Amazon S3

You can use Cloudera Manager to replicate Hive data and metadata to and from Amazon S3, however you cannot replicate data from one Amazon S3 instance to another using Cloudera Manager. You must have the appropriate credentials to access the Amazon S3 account and you must create buckets in Amazon S3 to store the replicated files.

When you replicate data to cloud storage with BDR, BDR backs up file metadata, including extended attributes and ACLs.

To configure Hive replication to or from Amazon S3:

1. Create an **External Account** using Amazon S3 credentials. See Configuring an External Account for Amazon S3 Replication.
2. Select **Backup** > **Replication Schedules**.
3. Click **Create Schedule** > **Hive Replication**
4. To back up data to S3:

   a. Select the Source cluster from the **Source** drop-down list.
   b. Select the Amazon S3 destination (one of the External Accounts you created for Amazon S3) from the **Destination** drop-down list.
   c. Enter the path where the data should be copied to in S3. Enter using the following form:

```
s3a://S3_bucket_name/path
```

   d. Select one of the following **Replication Options**:

      - **Metadata and Data** – Backs up the Hive data from HDFS and its associated metadata.
      - **Metadata only** – Backs up only the Hive metadata.

5. To restore data from S3:

   a. Select the Amazon S3 source (one of the External Accounts you created for Amazon S3) from the **Source** drop-down list.
   b. Select the destination cluster from the **Destination** drop-down list.
   c. Enter the path to the metadata file (`export.json`) where the data should be copied from in S3. Enter using the following form:

```
s3a://S3_bucket_name/path_to_metadata_file
```

   d. Select one of the following **Replication Options**:

      - **Metadata and Data** – Restores the Hive data from HDFS from S3 and its associated metadata.
      - **Metadata only** – Restores only the Hive metadata.
      - **Hive-on-S3** – Restores only the Hive tables and references the tables on S3 as a Hive external table. If you drop a table in Hive, the data remains on S3. Only data that was backed up using a Hive/Impala Replication schedule can be restored. However, you can restore a Hive external table that is stored in S3.

6. Complete the configuration of the Hive replication schedule by following the steps under Configuring Replication of Hive Data on page 77, beginning with step 5.c on page 78

Hive Authentication

Hive authentication involves configuring Hive metastore, HiveServer2, and all Hive clients to use your deployment of LDAP/Active Directory Kerberos on your cluster.

Here is a summary of the status of Hive authentication in CDH 5:

- HiveServer2 supports authentication of the Thrift client using Kerberos or user/password validation backed by LDAP. For configuration instructions, see HiveServer2 Security Configuration.
- Earlier versions of HiveServer do not support Kerberos authentication for clients. However, the Hive MetaStoreServer does support Kerberos authentication for Thrift clients. For configuration instructions, see Hive MetaStoreServer Security Configuration.

  See also: Using Hive to Run Queries on a Secure HBase Server on page 93

For authorization, Hive uses Apache Sentry to enable role-based, fine-grained authorization for HiveServer2. See Apache Sentry Overview.

> **Important:** Cloudera does not support Apache Ranger or Hive's native authorization frameworks for configuring access control in Hive. Use Cloudera-supported Apache Sentry instead.

HiveServer2 Security Configuration

HiveServer2 supports authentication of the Thrift client using the following methods:

- Kerberos authentication
- LDAP authentication

  Starting with CDH 5.7, clusters running LDAP-enabled HiveServer2 deployments also accept Kerberos authentication. This ensures that users are not forced to enter usernames/passwords manually, and are able to take advantage of the multiple authentication schemes SASL offers. In CDH 5.6 and lower, HiveServer2 stops accepting delegation tokens when any alternate authentication is enabled.

Kerberos authentication is supported between the Thrift client and HiveServer2, and between HiveServer2 and secure HDFS. LDAP authentication is supported only between the Thrift client and HiveServer2.

To configure HiveServer2 to use one of these authentication modes, configure the `hive.server2.authentication` configuration property.

Enabling Kerberos Authentication for HiveServer2

If you configure HiveServer2 to use Kerberos authentication, HiveServer2 acquires a Kerberos ticket during startup. HiveServer2 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.

Configuring HiveServer2 for Kerberos-Secured Clusters

To enable Kerberos Authentication for HiveServer2, add the following properties in the `/etc/hive/conf/hive-site.xml` file:

```
<property>
  <name>hive.server2.authentication</name>
  <value>KERBEROS</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.principal</name>
  <value>hive/_HOST@YOUR-REALM.COM</value>
</property>
<property>
  <name>hive.server2.authentication.kerberos.keytab</name>
  <value>/etc/hive/conf/hive.keytab</value>
</property>
```

where:

- `hive.server2.authentication` is a client-facing property that controls the type of authentication HiveServer2 uses for connections to clients. In this case, HiveServer2 uses Kerberos to authenticate incoming clients.
- The `_HOST@YOUR-REALM.COM` value in the example above is the Kerberos principal for the host where HiveServer2 is running. The string _HOST in the properties is replaced at run time by the fully qualified domain name (FQDN) of the host machine where the daemon is running. Reverse DNS must be working on all the hosts configured this way. Replace `YOUR-REALM.COM` with the name of the Kerberos realm your Hadoop cluster is in.
- The `/etc/hive/conf/hive.keytab` value in the example above is a keytab file for that principal.

If you configure HiveServer2 to use both Kerberos authentication and secure impersonation, JDBC clients and Beeline can specify an alternate session user. If these clients have proxy user privileges, HiveServer2 impersonates the alternate user instead of the one connecting. The alternate user can be specified by the JDBC connection string `proxyUser=`*`userName`*

Configuring JDBC Clients for Kerberos Authentication with HiveServer2 (Using the Apache Hive Driver in Beeline)

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/HiveServer2Host@YOUR-REALM.COM"
Connection con = DriverManager.getConnection(url);
```

where `hive` is the principal configured in `hive-site.xml` and `HiveServer2Host` is the host where HiveServer2 is running.

For JDBC clients using the **Cloudera JDBC driver**, see Cloudera JDBC Driver for Hive. For ODBC clients, see Cloudera ODBC Driver for Apache Hive.

Using Beeline to Connect to a Secure HiveServer2

Use the following command to start `beeline` and connect to a secure HiveServer2 process. In this example, the HiveServer2 process is running on `localhost` at port `10000`:

```
$ /usr/lib/hive/bin/beeline
beeline> !connect
jdbc:hive2://localhost:10000/default;principal=hive/HiveServer2Host@YOUR-REALM.COM
0: jdbc:hive2://localhost:10000/default>
```

For more information about the Beeline CLI, see Using the Beeline CLI.

For instructions on encrypting communication with the ODBC/JDBC drivers, see Configuring Encrypted Communication Between HiveServer2 and Client Drivers on page 95.

Using LDAP Username/Password Authentication with HiveServer2

As an alternative to Kerberos authentication, you can configure HiveServer2 to use user and password validation backed by LDAP. The client sends a username and password during connection initiation. HiveServer2 validates these credentials using an external LDAP service.

You can enable LDAP Authentication with HiveServer2 using Active Directory or OpenLDAP.

Important: When using LDAP username/password authentication with HiveServer2, you must enable encrypted communication between HiveServer2 and its client drivers to avoid sending cleartext passwords. For instructions, see Configuring Encrypted Communication Between HiveServer2 and Client Drivers on page 95. To avoid sending LDAP credentials over a network in cleartext, see Configuring LDAPS Authentication with HiveServer2 on page 89.

Enabling LDAP Authentication with HiveServer2 using Active Directory

- **For managed clusters, use Cloudera Manager:**

  1. In the Cloudera Manager Admin Console, click **Hive** in the list of components, and then select the **Configuration** tab.
  2. Type "ldap" in the Search text box to locate the LDAP configuration fields.
  3. Check **Enable LDAP Authentication**.
  4. Enter the **LDAP URL** in the format `ldap[s]://<host>:<port>`
  5. Enter the **Active Directory Domain** for your environment.
  6. Click **Save Changes**.

- **For unmanaged clusters, use the command line:**

  Add the following properties to the `hive-site.xml`:

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>LDAP_URLL</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.Domain</name>
  <value>AD_DOMAIN_ADDRESS</value>
</property>
```

Where:

The *`LDAP_URL`* value is the access URL for your LDAP server. For example, `ldap[s]://<host>:<port>`

Enabling LDAP Authentication with HiveServer2 using OpenLDAP

To enable LDAP authentication using OpenLDAP, include the following properties in `hive-site.xml`:

```
<property>
  <name>hive.server2.authentication</name>
  <value>LDAP</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.url</name>
  <value>LDAP_URL</value>
</property>
<property>
  <name>hive.server2.authentication.ldap.baseDN</name>
  <value>LDAP_BaseDN</value>
</property>
```

where:

- The `LDAP_URL` value is the access URL for your LDAP server.
- The `LDAP_BaseDN` value is the base LDAP DN for your LDAP server; for example, `ou=People,dc=example,dc=com`.

Configuring JDBC Clients for LDAP Authentication with HiveServer2

The JDBC client requires a connection URL as shown below.

JDBC-based clients must include `user=LDAP_Userid;password=LDAP_Password` in the JDBC connection string. For example:

```
String url
="jdbc:hive2://#<host>:#<port>/#<dbName>;ssl=true;sslTrustStore=#<ssl_truststore_path>;trustStorePassword=#<truststore_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.

For ODBC Clients, see Cloudera ODBC Driver for Apache Hive.

Enabling LDAP Authentication for HiveServer2 in Hue

Enable LDAP authentication with HiveServer2 by setting the following properties under the `[beeswax]` section in `hue.ini`.

`auth_username`	LDAP username of Hue user to be authenticated.
`auth_password`	LDAP password of Hue user to be authenticated.

Hive uses these login details to authenticate to LDAP. The Hive service trusts that Hue has validated the user being impersonated.

Configuring LDAPS Authentication with HiveServer2

HiveServer2 supports LDAP username/password authentication for clients. Clients send LDAP credentials to HiveServer2 which in turn verifies them against the configured LDAP provider, such as OpenLDAP or Microsoft Active Directory. Most implementations now support LDAPS (LDAP over TLS/SSL), an authentication protocol that uses TLS/SSL to encrypt communication between the LDAP service and its client (in this case, HiveServer2) to avoid sending LDAP credentials in cleartext.

To configure the LDAPS service with HiveServer2:

1. Import the LDAP server CA certificate or the server certificate into a truststore on the HiveServer2 host. If you import the CA certificate, HiveServer2 will trust any server with a certificate issued by the LDAP server's CA. If you only import the server certificate, HiveServer2 trusts only that server. See Creating Java Keystores and Truststores for more details.
2. Make sure the truststore file is readable by the `hive` user.
3. Set the `hive.server2.authentication.ldap.url` configuration property in `hive-site.xml` to the LDAPS URL. For example, `ldaps://sample.myhost.com`.

   > **Note:** The URL scheme should be `ldaps` and *not* `ldap`.

4. If this is a managed cluster, in Cloudera Manager, go to the Hive service and select **Configuration**. Under Category, select **Security**. In the right panel, search for **HiveServer2 TLS/SSL Certificate Trust Store File**, and add the path to the truststore file that you created in step 1.

   If you are using an unmanaged cluster, set the environment variable `HADOOP_OPTS` as follows:

```
HADOOP_OPTS="-Djavax.net.ssl.trustStore=<trustStore-file-path>
-Djavax.net.ssl.trustStorePassword=<trustStore-password>"
```

5. Restart HiveServer2.

Pluggable Authentication

Pluggable authentication allows you to provide a custom authentication provider for HiveServer2.

To enable pluggable authentication:

1. Set the following properties in `/etc/hive/conf/hive-site.xml`:

```
<property>
  <name>hive.server2.authentication</name>
  <value>CUSTOM</value>
  <description>Client authentication types.
  NONE: no authentication check
  LDAP: LDAP/AD based authentication
  KERBEROS: Kerberos/GSSAPI authentication
  CUSTOM: Custom authentication provider
  (Use with property hive.server2.custom.authentication.class)
  </description>
</property>

<property>
  <name>hive.server2.custom.authentication.class</name>
  <value>pluggable-auth-class-name</value>
  <description>
  Custom authentication class. Used when property
  'hive.server2.authentication' is set to 'CUSTOM'. Provided class
  must be a proper implementation of the interface
  org.apache.hive.service.auth.PasswdAuthenticationProvider. HiveServer2
  will call its Authenticate(user, passed) method to authenticate requests.
  The implementation may optionally extend the Hadoop's
  org.apache.hadoop.conf.Configured class to grab Hive's Configuration object.
  </description>
</property>
```

2. Make the class available in the CLASSPATH of HiveServer2.

Trusted Delegation with HiveServer2

HiveServer2 determines the identity of the connecting user from the underlying 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 at the Hadoop level, then all MapReduce jobs and HDFS accesses will be performed with the identity of the connecting user. If Apache Sentry is configured, then this connecting userid can also be used to verify access rights to underlying tables and views.

In CDH 4.5, a connecting user (for example, hue) with Hadoop-level superuser privileges, can request an alternate user for the given session. HiveServer2 will check if the connecting user has Hadoop-level privileges to proxy the requested userid (for example, bob). If it does, then the new session will be run on behalf of the alternate user, bob, requested by connecting user, hue.

To specify an alternate user for new connections, the JDBC client needs to add the `hive.server2.proxy.user=<alternate_user_id>` property to the JDBC connection URL. Note that the connecting user needs to have Hadoop-level proxy privileges over the alternate user. For example, if user hue requests access to run a session as user bob, the JDBC connection string should be as follows:

```
# Login as super user Hue
kinit hue -k -t hue.keytab hue@MY-REALM.COM

# Connect using following JDBC connection string
#
jdbc:hive2://myHost.myOrg.com:10000/default;principal=hive/_HOST@MY-REALM.COM;hive.server2.proxy.user=bob
```

HiveServer2 Impersonation

> **Important:** This is not the recommended method to implement HiveServer2 authorization. Cloudera recommends you use Sentry to implement this instead.

Impersonation in HiveServer2 allows users to execute queries and access HDFS files as the connected user rather than the super user who started the HiveServer2 daemon. This enforces an access control policy at the file level using HDFS file permissions or ACLs. Keeping impersonation enabled means Sentry does not have end-to-end control over the authorization process. While Sentry can enforce access control policies on tables and views in the Hive warehouse, it has no control over permissions on the underlying table files in HDFS. Hence, even if users do not have the Sentry privileges required to access a table in the warehouse, as long as they have permission to access the corresponding table file in HDFS, any jobs or queries submitted will bypass Sentry authorization checks and execute successfully.

To configure Sentry correctly, restrict ownership of the Hive warehouse to `hive:hive` and disable Hive impersonation as described here.

To enable impersonation in HiveServer2:

1. Add the following property to the `/etc/hive/conf/hive-site.xml` file and set the value to `true`. (The default value is `false`.)

```
<property>
  <name>hive.server2.enable.impersonation</name>
  <description>Enable user impersonation for HiveServer2</description>
  <value>true</value>
</property>
```

2. In HDFS or MapReduce configurations, add the following property to the `core-site.xml` file:

```
<property>
  <name>hadoop.proxyuser.hive.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hive.groups</name>
  <value>*</value>
</property>
```

See also File System Permissions.

Securing the Hive Metastore

> **Note:** This is not the recommended method to protect the Hive Metastore. Cloudera recommends you use Sentry to implement this instead.

To prevent users from accessing the Hive metastore and the Hive metastore database using any method other than through HiveServer2, the following actions are recommended:

- 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.
- Grant access to the metastore database only from the metastore service host. This is specified for MySQL as:

```
GRANT ALL PRIVILEGES ON metastore.* TO 'hive'@'metastorehost';
```

  where `metastorehost` is the host where the metastore service is running.

- Make sure users who are not admins cannot log on to the host on which HiveServer2 runs.

Disabling the Hive Security Configuration

Hive's security related metadata is stored in the configuration file `hive-site.xml`. The following sections describe how to disable security for the Hive service.

Disable Client/Server Authentication

To disable client/server authentication, set `hive.server2.authentication` to `NONE`. For example,

```
<property>
  <name>hive.server2.authentication</name>
  <value>NONE</value>
  <description>
    Client authentication types.
       NONE: no authentication check
       LDAP: LDAP/AD based authentication
       KERBEROS: Kerberos/GSSAPI authentication
       CUSTOM: Custom authentication provider
               (Use with property hive.server2.custom.authentication.class)
  </description>
</property>
```

Disable Hive Metastore security

To disable Hive Metastore security, perform the following steps:

- Set the `hive.metastore.sasl.enabled` property to `false` in all configurations, the metastore service side as well as for all clients of the metastore. For example, these might include HiveServer2, Impala, Pig and so on.
- Remove or comment the following parameters in `hive-site.xml` for the metastore service. Note that this is a server-only change.
  - `hive.metastore.kerberos.keytab.file`
  - `hive.metastore.kerberos.principal`

Disable Underlying Hadoop Security

If you also want to disable the underlying Hadoop security, remove or comment out the following parameters in `hive-site.xml`.

- `hive.server2.authentication.kerberos.keytab`
- `hive.server2.authentication.kerberos.principal`

Hive Metastore Server Security Configuration

> **Important:**
>
> This section describes how to configure security for the Hive metastore server. If you are using HiveServer2, see HiveServer2 Security Configuration.

Here is a summary of Hive metastore server security in CDH 5:

- No additional configuration is required to run Hive on top of a security-enabled Hadoop cluster in standalone mode using a local or embedded metastore.
- HiveServer does not support Kerberos authentication for clients. While it is possible to run HiveServer with a secured Hadoop cluster, doing so creates a security hole since HiveServer does not authenticate the Thrift clients that connect to it. Instead, you can use HiveServer2 HiveServer2 Security Configuration.
- The Hive metastore server supports Kerberos authentication for Thrift clients. For example, you can configure a standalone Hive metastore server instance to force clients to authenticate with Kerberos by setting the following properties in the `hive-site.xml` configuration file used by the metastore server:

```
<property>
  <name>hive.metastore.sasl.enabled</name>
  <value>true</value>
  <description>If true, the metastore thrift interface will be secured with SASL. Clients
 must authenticate with Kerberos.</description>
</property>
```

```
<property>
  <name>hive.metastore.kerberos.keytab.file</name>
  <value>/etc/hive/conf/hive.keytab</value>
  <description>The path to the Kerberos Keytab file containing the metastore thrift
server's service principal.</description>
</property>

<property>
  <name>hive.metastore.kerberos.principal</name>
  <value>hive/_HOST@YOUR-REALM.COM</value>
  <description>The service principal for the metastore thrift server. The special string
 _HOST will be replaced automatically with the correct host name.</description>
</property>
```

Note:

The values shown above for the `hive.metastore.kerberos.keytab.file` and `hive.metastore.kerberos.principal` properties are examples which you will need to replace with the appropriate values for your cluster. Also note that the Hive keytab file should have its access permissions set to `600` and be owned by the same account that is used to run the Metastore server, which is the `hive` user by default.

- Requests to access the metadata are fulfilled by the Hive metastore impersonating the requesting user. This includes read access to the list of databases, tables, properties of each table such as their HDFS location and file type. You can restrict access to the Hive metastore service by allowing it to impersonate only a subset of Kerberos users. This can be done by setting the `hadoop.proxyuser.hive.groups` property in `core-site.xml` on the Hive metastore host.

  For example, if you want to give the `hive` user permission to impersonate members of groups `hive` and `user1`:

```
<property>
<name>hadoop.proxyuser.hive.groups</name>
<value>hive,user1</value>
</property>
```

In this example, the Hive metastore can impersonate users belonging to *only* the `hive` and `user1` groups. Connection requests from users not belonging to these groups will be rejected.

Using Hive to Run Queries on a Secure HBase Server

To use Hive to run queries on a secure HBase Server, you must set the following `HIVE_OPTS` environment variable:

```
env HIVE_OPTS="-hiveconf hbase.security.authentication=kerberos -hiveconf
hbase.master.kerberos.principal=hbase/_HOST@YOUR-REALM.COM -hiveconf
hbase.regionserver.kerberos.principal=hbase/_HOST@YOUR-REALM.COM -hiveconf
hbase.zookeeper.quorum=zookeeper1,zookeeper2,zookeeper3" hive
```

where:

- You replace `YOUR-REALM` with the name of your Kerberos realm
- You replace `zookeeper1,zookeeper2,zookeeper3` with the names of your ZooKeeper servers. The `hbase.zookeeper.quorum` property is configured in the `hbase-site.xml` file.
- The special string _HOST is replaced at run-time by the fully qualified domain name of the host machine where the HBase Master or RegionServer is running. This requires that reverse DNS is properly working on all the hosts configured this way.

In the following, _HOST is the name of the host where the HBase Master is running:

```
-hiveconf hbase.master.kerberos.principal=hbase/_HOST@YOUR-REALM.COM
```

In the following, _HOST is the hostname of the HBase RegionServer that the application is connecting to:

```
-hiveconf hbase.regionserver.kerberos.principal=hbase/_HOST@YOUR-REALM.COM
```

Note:

You can also set the HIVE_OPTS environment variable in your shell profile.

Configuring Encrypted Communication Between HiveServer2 and Client Drivers

Starting with CDH 5.5, encryption for HiveServer2 clients has been decoupled from the authentication mechanism. This means you can use either SASL QOP or TLS/SSL to encrypt traffic between HiveServer2 and its clients, irrespective of whether Kerberos is being used for authentication. Previously, the JDBC client drivers only supported SASL QOP encryption on Kerberos-authenticated connections.

SASL QOP encryption is better suited for encrypting RPC communication and may result in performance issues when dealing with large amounts of data. Move to using TLS/SSL encryption to avoid such issues.

This topic describes how to set up encrypted communication between HiveServer2 and its JDBC/ODBC client drivers.

Configuring Encrypted Client/Server Communication Using TLS/SSL

You can use either the Cloudera Manager or the command-line instructions described below to enable TLS/SSL encryption for JDBC/ODBC client connections to HiveServer2. For background information on setting up TLS/SSL truststores and keystores, see TLS/SSL Certificates Overview.

Note: Cloudera Manager and CDH components support either TLS 1.0, TLS 1.1, or TLS 1.2, but not SSL 3.0. References to SSL continue only because of its widespread use in technical jargon.

Using Cloudera Manager

The steps for configuring and enabling TLS/SSL for Hive are as follows:

1. Open the Cloudera Manager Admin Console and go to the Hive service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Hive (Service-Wide)**.
4. Select **Category** > **Security**.
5. In the Search field, type **TLS/SSL** to show the Hive properties.
6. Edit the following properties according to your cluster configuration.

Table 3: Hive TLS/SSL Properties

Property	Description
Enable TLS/SSL for HiveServer2	Enable support for encrypted client-server communication using Transport Layer Security (TLS) for HiveServer2 connections.
HiveServer2 TLS/SSL Server JKS Keystore File Location	Path to the TLS keystore.
HiveServer2 TLS/SSL Server JKS Keystore File Password	Password for the TLS keystore.

7. Click **Save Changes** to commit the changes.
8. Restart the Hive service.

Using the Command Line

- To enable TLS/SSL, add the following configuration parameters to `hive-site.xml` :

```
<property>
  <name>hive.server2.use.SSL</name>
  <value>true</value>
  <description>enable/disable SSL </description>
</property>

<property>
  <name>hive.server2.keystore.path</name>
  <value>keystore-file-path</value>
  <description>path to keystore file</description>
</property>

<property>
  <name>hive.server2.keystore.password</name>
  <value>keystore-file-password</value>
  <description>keystore password</description>
</property>
```

- The `keystore` must contain the server's certificate.
- The JDBC client must add the following properties in the connection URL when connecting to a HiveServer2 using TLS/SSL:

```
;ssl=true[;sslTrustStore=<Trust-Store-Path>;trustStorePassword=<Trust-Store-password>]
```

- Make sure one of the following is true:
  - *Either:* `sslTrustStore` points to the truststore file containing the server's certificate; for example:

```
jdbc:hive2://localhost:10000/default;ssl=true;\
sslTrustStore=/home/usr1/ssl/trust_store.jks;trustStorePassword=xyz
```

  - *or:* the Trust Store arguments are set using the Java system properties `javax.net.ssl.trustStore` and `javax.net.ssl.trustStorePassword`; for example:

```
java -Djavax.net.ssl.trustStore=/home/usr1/ssl/trust_store.jks
-Djavax.net.ssl.trustStorePassword=xyz \
 MyClass jdbc:hive2://localhost:10000/default;ssl=true
```

For more information on using self-signed certificates and the Trust Store, see the Oracle Java SE keytool page.

Configuring Encrypted Client/Server Communication Using SASL QOP

Traffic between the Hive JDBC or ODBC drivers and HiveServer2 can be encrypted using plain SASL QOP encryption which allows you to preserve data integrity (using checksums to validate message integrity) and confidentiality (by encrypting messages). This can be enabled by setting the `hive.server2.thrift.sasl.qop` property in `hive-site.xml`. For example,

```
<property>
<name>hive.server2.thrift.sasl.qop</name>
<value>auth-conf</value>
<description>Sasl QOP value; one of 'auth', 'auth-int' and 'auth-conf'</description>
</property>
```

Valid settings for the `value` field are:

- `auth`: Authentication only (default)
- `auth-int`: Authentication with integrity protection

- `auth-conf`: Authentication with confidentiality protection

The parameter value that you specify above in the HiveServer2 configuration, should match that specified in the Beeline client connection JDBC URL. For example:

```
!connect jdbc:hive2://ip-10-5-15-197.us-west-2.compute.internal:10000/default;  \
principal=hive/_HOST@US-WEST-2.COMPUTE.INTERNAL;sasl.qop=auth-conf
```

Hive SQL Syntax for Use with Sentry

Sentry permissions can be configured through GRANT and REVOKE statements issued either interactively or programmatically through the HiveServer2 SQL command line interface, Beeline (documentation available here). The syntax described below is very similar to the GRANT and REVOKE commands that are available in well-established relational database systems.

In HUE, the Sentry Admin that creates roles and grants privileges must belong to a group that has ALL privileges on the server. For example, you can create a role for the group that contains the hive or impala user, and grant ALL ON SERVER .. WITH GRANT OPTION to that role:

```
CREATE ROLE <admin role>;
GRANT ALL ON SERVER <server1> TO ROLE <admin role> WITH GRANT OPTION;
GRANT ROLE <admin role> TO GROUP <hive>;
```

> **Important:**
>
> - When Sentry is enabled, you must use Beeline to execute Hive queries. Hive CLI is not supported with Sentry and must be disabled. See Disabling Hive CLI for information on how to disable the Hive CLI.
> - There are some differences in syntax between Hive and the corresponding Impala SQL statements. For Impala syntax, see SQL Statements.
> - No privilege is required to drop a function. Any user can drop a function.

Sentry supports column-level authorization with the SELECT privilege. Information about column-level authorization is in the Column-level Authorization on page 98 section of this page.

See the sections below for details about the supported statements and privileges:

Column-level Authorization

Sentry allows you to assign the SELECT privilege on a subset of columns in a table.

The following command grants a role the SELECT privilege on a column:

```
GRANT SELECT <column name> ON TABLE <table name> TO ROLE <role name>;
```

The following command can be used to revoke the SELECT privilege on a column:

```
REVOKE SELECT <column name> ON TABLE <table name> FROM ROLE <role name>;
```

Any new columns added to a table will be inaccessible by default, until explicitly granted access.

Actions allowed for users with SELECT privilege on a column:

Users whose roles have been granted the SELECT privilege on columns only, can perform operations which explicitly refer to those columns. Some examples are:

```
SELECT <column name> FROM TABLE <table name>;
```

In this case, Sentry will first check to see if the user has the required privileges to access the table. It will then further check to see whether the user has the SELECT privilege to access the column(s).

```
SELECT COUNT <column name> FROM TABLE <table name>;
```

Users are also allowed to use the COUNT function to return the number of values in the column.

```
SELECT <column name> FROM TABLE <table name> WHERE <column name> <operator> GROUP BY
<column name>;
```

The above command will work as long as you refer only to columns to which you already have access.

- To list the column(s) to which the current user has SELECT access:

```
SHOW COLUMNS (FROM|IN) <table name> [(FROM|IN) <database name>];
```

Exceptions:

- If a user has SELECT access to all columns in a table, the following command will work. Note that this is an exception, not the norm. In all other cases, SELECT on all columns does *not* allow you to perform table-level operations.

```
SELECT * FROM TABLE <table name>;
```

Limitations:

- Column-level privileges can only be applied to tables, not partitions or views.
- **HDFS-Sentry Sync:** With HDFS-Sentry sync enabled, even if a user has been granted access to all columns of a table, they will not have access to the corresponding HDFS data files. This is because Sentry does not consider SELECT on all columns equivalent to explicitly being granted SELECT on the table.
- Column-level access control for access from Spark SQL is not supported by the HDFS-Sentry plug-in.

CREATE ROLE Statement

The CREATE ROLE statement creates a role to which privileges can be granted. Privileges can be granted to roles, which can then be assigned to users. A user that has been assigned a role will only be able to exercise the privileges of that role.

Only users that have administrative privileges can create or drop roles. By default, the hive, impala and hue users have admin privileges in Sentry.

```
CREATE ROLE <role name>;
```

Note that role names are case-insensitive.

DROP ROLE Statement

The DROP ROLE statement can be used to remove a role from the database. Once dropped, the role will be revoked for all users to whom it was previously assigned. Queries that are already executing will not be affected. However, since Hive checks user privileges before executing each query, active user sessions in which the role has already been enabled will be affected.

```
DROP ROLE <role name>;
```

GRANT ROLE Statement

The `GRANT ROLE` statement can be used to grant roles to groups. Only Sentry admin users can grant roles to a group.

```
GRANT ROLE <role name> [, <role name>]
    TO GROUP <group name> [,GROUP <group name>]
```

Sentry only allows you to grant roles to groups that have alphanumeric characters and underscores (_) in the group name. If the group name contains a non-alphanumeric character that is not an underscore, you can put the group name in backticks (`) to execute the command. For example, Sentry will return an error for the following command:

```
GRANT ROLE test TO GROUP test-group;
```

To grant a role to this group, put the group name in backticks:

```
GRANT ROLE test TO GROUP `test-group`;
```

The following command, which contains an underscore, is also acceptable:

```
GRANT ROLE test TO GROUP test_group;
```

Operating system group names must be in lowercase letters. Although group names are case-insensitive to Sentry, Sentry modifies capital letters within group names to be lowercase. For example, Sentry will change `TestGroup` to `testgroup`. It is not possible to disable this normalization. Therefore, group information within the environment must be in lowercase letters.

REVOKE ROLE Statement

The `REVOKE ROLE` statement can be used to revoke roles from groups. Only Sentry admin users can revoke the role from a group.

```
REVOKE ROLE <role name> [, <role name>]
    FROM GROUP <group name> [,GROUP <group name>]
```

GRANT <Privilege> Statement

Use the GRANT <Privilege> statement to grant privileges on an object to a role. The statement uses the following syntax:

```
GRANT
    <privilege> [, <privilege> ]
    ON <object type> <object name>
    TO ROLE <role name> [,ROLE <role name>]
```

You can grant the `SELECT` privilege on specific columns of a table. For example:

```
GRANT SELECT <column name> ON TABLE <table name> TO ROLE <role name>;
```

GRANT <Privilege> ON URIs (HDFS and S3A)

If the GRANT for Sentry URI does not specify the complete scheme, or the URI mentioned in Hive DDL statements does not have a scheme, Sentry automatically completes the URI by applying the default scheme based on the HDFS configuration provided in the `fs.defaultFS` property. Using the same HDFS configuration, Sentry can also auto-complete URIs in case the URI is missing a scheme and an authority component.

When a user attempts to access a URI, Sentry will check to see if the user has the required privileges. During the authorization check, if the URI is incomplete, Sentry will complete the URI using the default HDFS scheme. Note that Sentry does not check URI schemes for completion when they are being used to grant privileges. This is because users can `GRANT` privileges on URIs that do not have a complete scheme or do not already exist on the filesystem.

For example, in CDH 5.8 and later, the following `CREATE EXTERNAL TABLE` statement works even though the statement does not include the URI scheme.

```
GRANT ALL ON URI 'hdfs://namenode:XXX/path/to/table' TO ROLE <role name>;
CREATE EXTERNAL TABLE foo LOCATION 'namenode:XXX/path/to/table' TO ROLE <role name>;
```

Similarly, the following `CREATE EXTERNAL TABLE` statement works even though it is missing scheme and authority components.

```
GRANT ALL ON URI 'hdfs://namenode:XXX/path/to/table' TO ROLE <role name>;
CREATE EXTERNAL TABLE foo LOCATION
  '/path/to/table'
```

Since Sentry supports both HDFS and Amazon S3, in CDH 5.8 and later, Cloudera recommends that you specify the fully qualified URI in `GRANT` statements to avoid confusion. If the underlying storage is a mix of S3 and HDFS, the risk of granting the wrong privileges increases. The following are examples of fully qualified URIs:

- **HDFS:** *`hdfs://host:port/path/to/hdfs/table`*
- **S3:** *`s3a://host:port/path/to/s3/table`*

REVOKE <Privilege> Statement

You can use the `REVOKE <Privilege>` statement to revoke previously-granted privileges that a role has on an object.

```
REVOKE
    <privilege> [, <privilege> ]
    ON <object type> <object name>
    FROM ROLE <role name> [,ROLE <role name>]
```

For example, you can revoke previously-granted `SELECT` privileges on specific columns of a table with the following statement:

```
REVOKE SELECT <column name> ON TABLE <table name> FROM ROLE <role name>;
```

GRANT <Privilege> ... WITH GRANT OPTION

You can delegate granting and revoking privileges to other roles. For example, a role that is granted a privilege `WITH GRANT OPTION` can `GRANT/REVOKE` the same privilege to/from other roles. Hence, if a role has the `ALL` privilege on a database and the `WITH GRANT OPTION` set, users granted that role can execute `GRANT/REVOKE` statements only for that database or child tables of the database.

```
GRANT
    <privilege>
    ON <object type> <object name>
    TO ROLE <role name>
    WITH GRANT OPTION
```

Only a role with `GRANT` option on a specific privilege or its parent privilege can revoke that privilege from other roles. Once the following statement is executed, all privileges with and without grant option are revoked.

```
REVOKE
    <privilege>
    ON <object type> <object name>
    FROM ROLE <role name>
```

Hive does not currently support revoking only the `WITH GRANT OPTION` from a privilege previously granted to a role. To remove the `WITH GRANT OPTION`, revoke the privilege and grant it again without the `WITH GRANT OPTION` flag.

SET ROLE Statement

Sentry enforces restrictions on queries based on the roles and privileges that the user has. A user can have multiple roles and a role can have multiple privileges.

The `SET ROLE` command enforces restrictions at the role level, not at the user level. When you use the `SET ROLE` command to make a role active, the role becomes current for the session. If a role is not current for the session, it is inactive and the user does not have the privileges assigned to that role. A user can only use the `SET ROLE` command for roles that have been granted to the user.

To list the roles that are current for the user, use the `SHOW CURRENT ROLES` command. By default, all roles that are assigned to the user are current.

You can use the following `SET ROLE` commands:

SET ROLE NONE

Makes all roles for the user inactive. When no role is current, the user does not have any privileges and cannot execute a query.

SET ROLE ALL

Makes all roles that have been granted to the user active. All privileges assigned to those roles are applied. When the user executes a query, the query is filtered based on those privileges.

SET ROLE *role name*

Makes a single role active. The privileges assigned to that role are applied. When the user executes a query, the query is filtered based on the privileges assigned to that role.

SHOW Statement

- To list the database(s) for which the current user has database, table, or column-level access:

```
SHOW DATABASES;
```

- To list the table(s) for which the current user has table or column-level access:

```
SHOW TABLES;
```

- To list the column(s) to which the current user has `SELECT` access:

```
SHOW COLUMNS (FROM|IN) <table name> [(FROM|IN) <database name>];
```

- To list all the roles in the system (only for sentry admin users):

```
SHOW ROLES;
```

- To list all the roles in effect for the current user session:

```
SHOW CURRENT ROLES;
```

- To list all the roles assigned to the given *`group name`* (only allowed for Sentry admin users and others users that are part of the group specified by *`group name`*):

```
SHOW ROLE GRANT GROUP group name;
```

- The SHOW statement can also be used to list the privileges that have been granted to a role or all the grants given to a role for a particular object.

  To list all the grants for the given <*role name*> (only allowed for Sentry admin users and other users that have been granted the role specified by <*role name*>). The following command will also list any column-level privileges:

```
SHOW GRANT ROLE <role name>;
```

- To list all the grants for a role on the given <*object name*> (only allowed for Sentry admin users and other users that have been granted the role specified by <*role name*>). The following command will also list any column-level privileges:

```
SHOW GRANT ROLE <role name> on <object type> <object name>;
```

Example: Using Grant/Revoke Statements to Match an Existing Policy File

Note: In the following example(s), `server1` refers to an alias Sentry uses for the associated Hive service. It does not refer to any physical server. This alias can be modified using the `hive.sentry.server` property in hive-site.xml. If you are using Cloudera Manager, modify the Hive property, **Server Name for Sentry Authorization**, in the **Service-Wide** > **Advanced** category.

Here is a sample policy file:

```
[groups]
# Assigns each Hadoop group to its set of roles
manager = analyst_role, junior_analyst_role
analyst = analyst_role
jranalyst = junior_analyst_role
customers_admin = customers_admin_role
admin = admin_role

[roles] # The URIs below define a landing skid which
# the user can use to import or export data from the system.
# Since the server runs as the user "hive" files in that directory
# must either have the group hive and read/write set or
# be world read/write.
analyst_role = server=server1->db=analyst1, \
    server=server1->db=jranalyst1->table=*->action=select
    server=server1->uri=hdfs://ha-nn-uri/landing/analyst1
junior_analyst_role = server=server1->db=jranalyst1, \
    server=server1->uri=hdfs://ha-nn-uri/landing/jranalyst1

# Implies everything on server1.
admin_role = server=server1
```

The following sections show how you can use the new GRANT statements to assign privileges to roles (and assign roles to groups) to match the sample policy file above.

Grant privileges to `analyst_role`:

```
CREATE ROLE analyst_role;
GRANT ALL ON DATABASE analyst1 TO ROLE analyst_role;
GRANT SELECT ON DATABASE jranalyst1 TO ROLE analyst_role;
GRANT ALL ON URI 'hdfs://ha-nn-uri/landing/analyst1' \
TO ROLE analyst_role;
```

Grant privileges to `junior_analyst_role`:

```
CREATE ROLE junior_analyst_role;
GRANT ALL ON DATABASE jranalyst1 TO ROLE junior_analyst_role;
GRANT ALL ON URI 'hdfs://ha-nn-uri/landing/jranalyst1' \
TO ROLE junior_analyst_role;
```

Grant privileges to `admin_role`:

```
CREATE ROLE admin_role;
GRANT ALL ON SERVER server1 TO ROLE admin_role;
```

Grant roles to groups:

```
GRANT ROLE admin_role TO GROUP admin;
GRANT ROLE analyst_role TO GROUP analyst;
GRANT ROLE jranalyst_role TO GROUP jranalyst;
```

Troubleshooting Hive

This section provides guidance on problems you may encounter while installing, upgrading, or running Hive.

With Hive, the most common troubleshooting aspects involve performance issues and managing disk space. Because Hive uses an underlying compute mechanism such as MapReduce or Spark, sometimes troubleshooting requires diagnosing and changing configuration in those lower layers.

HiveServer2 Performance Tuning and Troubleshooting

HiveServer2 (HS2) services might require more memory if there are:

- Many Hive table partitions.
- Many concurrent connections to HS2.
- Complex Hive queries that access significant numbers of table partitions.

If any of these conditions exist, Hive can run slowly or possibly crash because the entire HS2 heap memory is full. This section describes the symptoms that occur when HS2 needs additional memory, how you can troubleshoot issues to identify their causes, and then address them.

Symptoms Displayed When HiveServer2 Heap Memory is Full

When HS2 heap memory is full, you might experience the following issues:

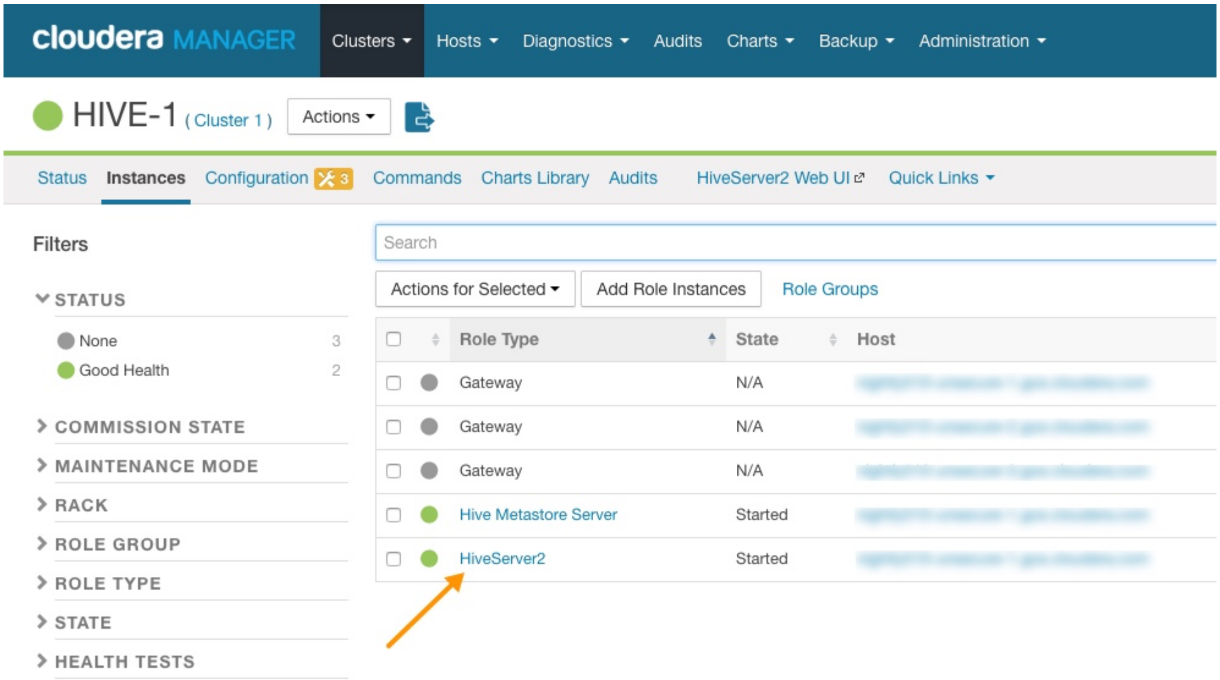
- HS2 service goes down and new sessions fail to start.
- HS2 service seems to be running fine, but client connections are refused.
- Query submission fails repeatedly.
- HS2 performance degrades and displays the following behavior:
  - Query submission delays
  - Long query execution times

Troubleshooting

HiveServer2 Service Crashes

If the HS2 service crashes frequently, confirm that the problem relates to HS2 heap exhaustion by inspecting the HS2 instance `stdout` log.

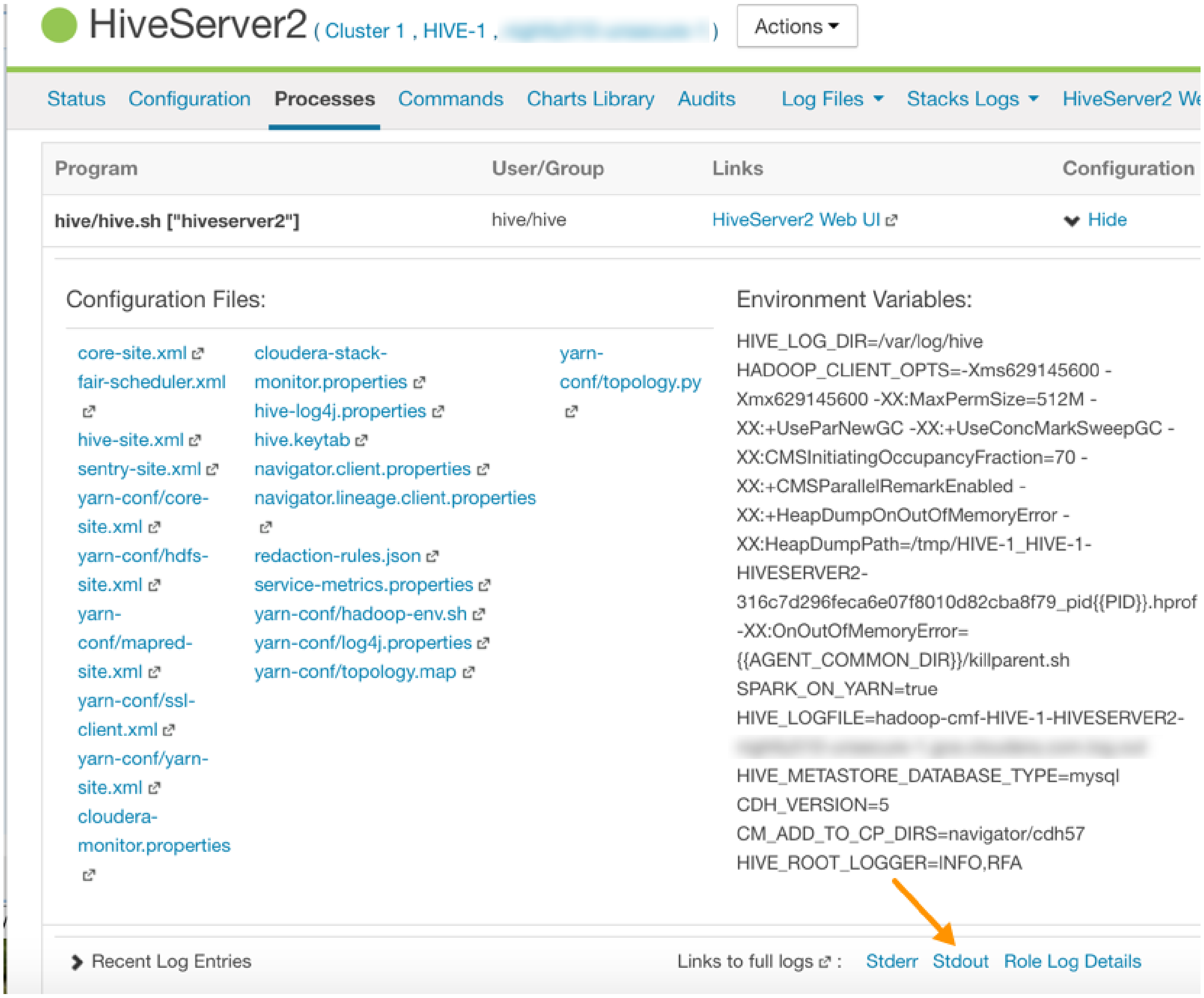
1. In Cloudera Manager, from the home page, go to **Hive > Instances**.
2. In the Instances page, click the link of the HS2 node that is down:



□		Role Type	State	Host
□	●	Gateway	N/A	[redacted]
□	●	Gateway	N/A	[redacted]
□	●	Gateway	N/A	[redacted]
□	●	Hive Metastore Server	Started	[redacted]
□	●	HiveServer2	Started	[redacted]

Figure 8: HiveServer2 Link on the Cloudera Manager Instances Page

3. On the HiveServer2 page, click **Processes**.
4. On the HiveServer2 Processes page, scroll down to the **Recent Log Entries** and click the link to the **Stdout** log.



HiveServer2 (Cluster 1 , HIVE-1 , [redacted])

Program	User/Group	Links	Configuration
hive/hive.sh ["hiveserver2"]	hive/hive	HiveServer2 Web UI	Hide

Configuration Files:

core-site.xml
fair-scheduler.xml
hive-site.xml
sentry-site.xml
yarn-conf/core-site.xml
yarn-conf/hdfs-site.xml
yarn-conf/mapred-site.xml
yarn-conf/ssl-client.xml
yarn-conf/yarn-site.xml
cloudera-monitor.properties
cloudera-stack-monitor.properties
hive-log4j.properties
hive.keytab
navigator.client.properties
navigator.lineage.client.properties
redaction-rules.json
service-metrics.properties
yarn-conf/hadoop-env.sh
yarn-conf/log4j.properties
yarn-conf/topology.map
yarn-conf/topology.py

Environment Variables:

```
HIVE_LOG_DIR=/var/log/hive
HADOOP_CLIENT_OPTS=-Xms629145600 -
Xmx629145600 -XX:MaxPermSize=512M -
XX:+UseParNewGC -XX:+UseConcMarkSweepGC -
XX:CMSInitiatingOccupancyFraction=70 -
XX:+CMSParallelRemarkEnabled -
XX:+HeapDumpOnOutOfMemoryError -
XX:HeapDumpPath=/tmp/HIVE-1_HIVE-1-
HIVESERVER2-
316c7d296feca6e07f8010d82cba8f79_pid{{PID}}.hprof
-XX:OnOutOfMemoryError=
{{AGENT_COMMON_DIR}}/killparent.sh
SPARK_ON_YARN=true
HIVE_LOGFILE=hadoop-cmf-HIVE-1-HIVESERVER2-
[redacted]
HIVE_METASTORE_DATABASE_TYPE=mysql
CDH_VERSION=5
CM_ADD_TO_CP_DIRS=navigator/cdh57
HIVE_ROOT_LOGGER=INFO,RFA
```

Figure 9: Link to the Stdout Log on the Cloudera Manager Processes Page

5. In the `stdout.log`, look for the following error:

```
# java.lang.OutOfMemoryError: Java heap space
# -XX:OnOutOfMemoryError="/usr/lib64/cmf/service/common/killparent.sh"
# Executing /bin/sh -c "/usr/lib64/cmf/service/common/killparent.sh"
```

Video: Troubleshooting HiveServer2 Service Crashes

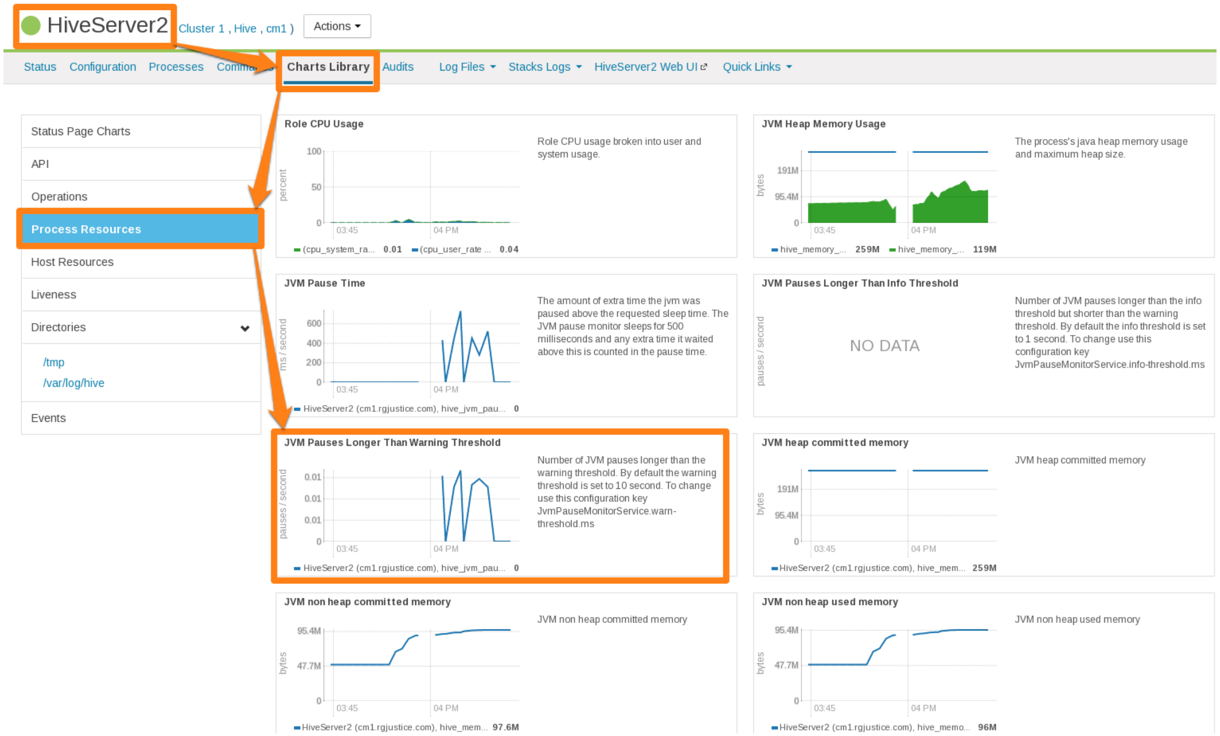
For more information about configuring Java heap size for HiveServer2, see the following video:

After you start the video, click **YouTube** in the lower right corner of the player window to watch it on YouTube where you can resize it for clearer viewing.

HiveServer2 General Performance Problems or Connections Refused

For general HS2 performance problems or if the service refuses connections, but does not completely hang, inspect the Cloudera Manager process charts:

1. In Cloudera Manager, navigate to **Home > Hive > Instances > HiveServer2 > Charts Library**.
2. In the **Process Resources** section of the Charts Library page, view the **JVM Pause Time** and the **JVM Pauses Longer Than Warning Threshold** charts for signs that JVM has paused to manage resources. For example:



Role CPU usage broken into user and system usage.

The process's java heap memory usage and maximum heap size.

The amount of extra time the jvm was paused above the requested sleep time. The JVM pause monitor sleeps for 500 milliseconds and any extra time it waited above this is counted in the pause time.

Number of JVM pauses longer than the info threshold but shorter than the warning threshold. By default the info threshold is set to 1 second. To change use this configuration key JvmPauseMonitorService.info-threshold.ms

Number of JVM pauses longer than the warning threshold. By default the warning threshold is set to 10 second. To change use this configuration key JvmPauseMonitorService.warn-threshold.ms

JVM heap committed memory

JVM non heap committed memory

JVM non heap used memory

Figure 10: Cloudera Manager Chart Library Page for Process Resources

HiveServer2 Performance Best Practices

High heap usage by the HS2 process can be caused by Hive queries accessing high numbers of table partitions (greater than several thousand), high levels of concurrency, or other Hive workload characteristics described in Identify Workload Characteristics That Increase Memory Pressure on page 60.

HiveServer2 Heap Size Configuration Best Practices

Optimal HS2 heap size configuration depends on several factors, including workload characteristics, number of concurrent clients, and the partitioning of underlying Hive tables. To resolve HS2 memory-related issues, confirm that the HS2 heap size is set properly for your environment.

1. In CDH 5.7 and higher, Cloudera Manager starts the HS2 service with 4 GB heap size by default unless hosts have insufficient memory. However, the heap size on lower versions of CDH or upgraded clusters might not be set to this recommended value. To raise the heap size to at least 4 GB:

   a. In Cloudera Manager, go to **Home** > **Hive** > **Configuration** > **HiveServer2** > **Resource Management**.
   b. Set **Java Heap Size of HiveServer2 in Bytes** to `4 GiB` and click **Save Changes**.
   c. From the **Actions** drop-down menu, select `Restart` to restart the HS2 service.

   If HS2 is already configured to run with 4 GB or greater heap size and there are still performance issues, workload characteristics may be causing memory pressure. Increase heap size to reduce memory pressure on HS2. Cloudera does not recommend exceeding 16 GB per instance because of long garbage collection pause times. See Identify Workload Characteristics That Increase Memory Pressure on page 60 for tips to optimize query workloads to reduce the memory requirements on HS2. Cloudera recommends splitting HS2 into multiple instances and load-balancing once you start allocating over 16 GB to HS2.

2. If workload analysis does not reveal any major issues, or you can only address workload issues over time, consider the following options:

   - Increase the heap size on HS2 in incremental steps. Cloudera recommends increasing the heap size by 50% from the current value with each step. If you have increased the heap size to 16 GB and issues persist, contact Cloudera Support.

- Reduce the number of services running on the HS2 host.
- Load-balance workloads across multiple HS2 instances as described in How the Number of Concurrent Connections Affect HiveServer2 Performance on page 60.
- Add more physical memory to the host or upgrade to a larger server.

How the Number of Concurrent Connections Affect HiveServer2 Performance

The number of concurrent connections can impact HS2 in the following ways:

- **High number of concurrent queries**

  High numbers of concurrent queries increases the connection count. Each query connection consumes resources for the query plan, number of table partitions accessed, and partial result sets. Limiting the number of concurrent users can help reduce overall HS2 resource consumption, especially limiting scenarios where one or more "in-flight" queries returns large result sets.

  How to resolve:

  - Load-balance workloads across multiple HS2 instances by using HS2 load balancing, which is available in CDH 5.7 and later. Cloudera recommends that you determine the total number of HS2 servers on a cluster by dividing the expected maximum number of concurrent users on a cluster by 40. For example, if 400 concurrent users are expected, 10 HS2 instances should be available to support them. See Configuring HiveServer2 High Availability in CDH on page 71 for setup instructions.
  - Review usage patterns, such as batch jobs timing or Oozie workflows, to identify spikes in the number of connections that can be spread over time.

- **Many abandoned Hue sessions**

  Users opening numerous browser tabs in Hue causes multiple sessions and connections. In turn, all of these open connections lead to multiple operations and multiple result sets held in memory for queries that finish processing. Eventually, this situation leads to a resource crisis.

  How to resolve:

  - Reduce the session timeout duration for HS2, which minimizes the impact of abandoned Hue sessions. To reduce session timeout duration, modify these configuration parameters as follows:

    - `hive.server2.idle.operation.timeout=7200000`

      The default setting for this parameter is `21600000` or 6 hours.

    - `hive.server2.idle.session.timeout=21600000`

      The default setting for this parameter is `43200000` or 12 hours.

      To set these parameters in Cloudera Manager, go to **Home > Hive > Configuration > HiveServer2 > Advanced** , and then search for each parameter.

  - Reduce the size of the result set returned by adding filters to queries. This minimizes memory pressure caused by "dangling" sessions.

Identify Workload Characteristics That Increase Memory Pressure

If increasing the heap size based on configuration guidelines does not improve performance, analyze your query workloads to identify characteristics that increase memory pressure on HS2. Workloads with the following characteristics increase memory requirements for HS2:

- **Queries that access a large number of table partitions:**

  - Cloudera recommends that a single query access no more than 10,000 table partitions. If joins are also used in the query, calculate the combined partition count accessed across all tables.

- Look for queries that load all table partitions in memory to execute. This can substantially add to memory pressure. For example, a query that accesses a partitioned table with the following SELECT statement loads all partitions of the target table to execute:

```
SELECT * FROM <table_name> LIMIT 10;
```

How to resolve:

- Add partition filters to queries to reduce the total number of partitions that are accessed. To view all of the partitions processed by a query, run the EXPLAIN DEPENDENCY clause, which is explained in the Apache Hive Language Manual.
- Set the `hive.metastore.limit.partition.request` parameter to 1000 to limit the maximum number of partitions accessed from a single table in a query. See the Apache wiki for information about setting this parameter. If this parameter is set, queries that access more than 1000 partitions fail with the following error:

```
MetaException: Number of partitions scanned (=%d) on table '%s' exceeds limit (=%d)
```

Setting this parameter protects against bad workloads and identifies queries that need to be optimized. To resolve the failed queries:

- Apply the appropriate partition filters.
- Override the limit on a per-query basis.
- Increase the cluster-wide limit beyond 1000, if needed, but note that this adds memory pressure to HiveServer2 and the Hive metastore.

- If the accessed table is not partitioned, see this Cloudera Engineering Blog post, which explains how to partition Hive tables to improve query performance. Choose columns or dimensions for partitioning based upon usage patterns. Partitioning tables too much causes data fragmentation, but partitioning too little causes queries to read too much data. Either extreme makes querying inefficient. Typically, a few thousand table partitions is fine.

- **Wide tables or columns:**

  - Memory requirements are directly proportional to the number of columns and the size of the individual columns. Typically, a wide table contains over 1,000 columns. Wide tables or columns can cause memory pressure if the number of columns is large. This is especially true for Parquet files because all data for a row-group must be in memory before it can be written to disk. Avoid wide tables when possible.
  - Large individual columns also cause the memory requirements to increase. Typically, this happens when a column contains free-form text or complex types.

  How to resolve:

  - Reduce the total number of columns that are materialized. If only a subset of columns are required, avoid `SELECT *` because it materializes all columns.
  - Instead, use a specific set of columns. This is particularly efficient for wide tables that are stored in column formats. Specify columns explicitly instead of using `SELECT *`, especially for production workloads.

- **High query complexity**

  Complex queries usually have large numbers of joins, often over 10 joins per query. HS2 heap size requirements increase significantly as the number of joins in a query increases.

  How to resolve:

  - Make sure that partition filters are specified on all partitioned tables that are involved in JOINs.
  - Whenever possible, break queries into multiple smaller queries with intermediate temporary tables.

- **Improperly written user-defined functions (UDFs)**

Improperly written UDFs can exert significant memory pressure on HS2.

How to resolve:

- Understand the memory implications of the UDF and test it before using it in production environments.

General Best Practices

The following general best practices help maintain a healthy Hive cluster:

- Review and test queries in a development or test cluster before running them in a production environment. Monitor heap memory usage while testing.
- Redirect and isolate any untested, unreviewed, ad-hoc, or "dangerous" queries to a separate HS2 instance that is not critical to batch operation.

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```