

cloudera[®]

Apache Hive Guide

Important Notice

© 2010-2021 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, and any other product or service names or slogans contained in this document are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder. If this documentation includes code, including but not limited to, code examples, Cloudera makes this available to you under the terms of the Apache License, Version 2.0, including any required notices. A copy of the Apache License Version 2.0, including any notices, is included herein. A copy of the Apache License Version 2.0 can also be found here: <https://opensource.org/licenses/Apache-2.0>

Hadoop and the Hadoop elephant logo are trademarks of the Apache Software Foundation. All other trademarks, registered trademarks, product names and company names or logos mentioned in this document are the property of their respective owners. Reference to any products, services, processes or other information, by trade name, trademark, manufacturer, supplier or otherwise does not constitute or imply endorsement, sponsorship or recommendation thereof by us.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Cloudera.

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property. For information about patents covering Cloudera products, see <http://tiny.cloudera.com/patents>.

The information in this document is subject to change without notice. Cloudera shall not be liable for any damages resulting from technical errors or omissions which may be present in this document, or from use of this document.

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 6.2.x
Date: August 2, 2021

Table of Contents

Best Practices for Using Apache Hive in CDH.....	8
Overview of Apache Hive Installation and Upgrade in CDH.....	9
Configuring Apache Hive in CDH.....	10
Configuring the Hive Metastore for CDH.....	10
<i>Metastore Deployment Modes.....</i>	<i>10</i>
<i>Supported Metastore Databases.....</i>	<i>11</i>
<i>Metastore Memory and Hardware Requirements.....</i>	<i>12</i>
<i>General Metastore Tuning Recommendations.....</i>	<i>13</i>
<i>Configuring the Metastore Database.....</i>	<i>13</i>
Configuring HiveServer2 for CDH.....	23
<i>HiveServer2 Memory and Hardware Requirements.....</i>	<i>23</i>
<i>hive.zookeeper.client.port.....</i>	<i>24</i>
<i>JDBC driver.....</i>	<i>25</i>
<i>Authentication.....</i>	<i>25</i>
<i>Running HiveServer2.....</i>	<i>25</i>
Starting the Hive Metastore in CDH.....	25
Apache Hive File System Permissions in CDH.....	26
Starting, Stopping, and Using HiveServer2 in CDH.....	26
<i>Using the Beeline CLI.....</i>	<i>27</i>
Using Apache Hive with HBase in CDH.....	27
Using the Metastore Schema Tool in CDH.....	28
<i>Schema Version Verification and Validation.....</i>	<i>28</i>
<i>Using schematool.....</i>	<i>29</i>
Installing Cloudera JDBC and ODBC Drivers on Clients in CDH.....	31
<i>Cloudera Hive JDBC Driver Download.....</i>	<i>31</i>
<i>Cloudera Hive ODBC Driver Download.....</i>	<i>32</i>
Setting HADOOP_MAPRED_HOME.....	32
Configuring the Hive Metastore to Use HDFS High Availability in CDH.....	32
Using & Managing Apache Hive in CDH.....	34
Hive Roles.....	34
Hive Execution Engines.....	34
Use Cases for Hive.....	35

Managing Hive Using Cloudera Manager.....	35
Overview of Ingesting and Querying Data with Apache Hive in CDH.....	36
<i>Ingesting Data with Hive.....</i>	36
<i>Column and Table Statistics for Query Optimization.....</i>	36
<i>Transaction (ACID) Support in Hive.....</i>	36
<i>Upstream Information for Hive.....</i>	36
Apache Parquet Tables with Hive in CDH.....	37
<i>Using Parquet Tables in Hive.....</i>	37
Running Apache Hive on Spark in CDH.....	38
<i>Configuring Hive on Spark.....</i>	38
<i>Dynamic Partition Pruning for Hive Map Joins.....</i>	39
<i>Using Hive UDFs with Hive on Spark.....</i>	44
<i>Troubleshooting Hive on Spark.....</i>	44
Using HiveServer2 Web UI in CDH.....	46
<i>Accessing the HiveServer2 Web UI.....</i>	46
<i>HiveServer2 Web UI Configuration.....</i>	46
Accessing Apache Hive Table Statistics in CDH.....	47
Managing Apache Hive User-Defined Functions.....	47
<i>Registering a UDF in Hive.....</i>	47
<i>Direct JAR Reference Configuration.....</i>	49
<i>Hive Aux JARs Directory Configuration.....</i>	49
<i>Reloadable Aux JAR Configuration.....</i>	49
<i>Creating Temporary Functions.....</i>	50
<i>Updating a User-Defined Function.....</i>	50
<i>Calling a Hive UDF from Impala.....</i>	50
<i>Adding Built-in UDFs to the HiveServer2 Blacklist.....</i>	51
Configuring Transient Apache Hive ETL Jobs to Use the Amazon S3 Filesystem in CDH.....	51
<i>About Transient Jobs.....</i>	52
<i>Configuring and Running Jobs on Transient Clusters.....</i>	52
Configuring a Shared Amazon RDS as an HMS for CDH.....	54
<i>Advantages of This Approach.....</i>	55
<i>How To Configure Amazon RDS as the Backend Database for a Shared Hive Metastore.....</i>	55
<i>Supported Scenarios.....</i>	55
Configuring ADLS Gen1 Connectivity.....	56
<i>Setting up ADLS to Use with CDH.....</i>	56
<i>Testing and Using ADLS Access.....</i>	57
<i>User-Supplied Key for Each Job.....</i>	57
<i>Single Master Key for Cluster-Wide Access.....</i>	58
<i>User-Supplied Key stored in a Hadoop Credential Provider.....</i>	58
<i>Create a Hadoop Credential Provider and reference it in a customized copy of the core-site.xml file for the service.....</i>	59
<i>Creating a Credential Provider for ADLS.....</i>	60
<i>ADLS Configuration Notes.....</i>	60
Importing Data into Hive with Sqoop Through HiverServer2.....	61

<i>Importing Data Through Hiveserver2</i>	61
<i>Importing Data</i>	61
<i>Steps</i>	61

Tuning Apache Hive in CDH.....63

Heap Size and Garbage Collection for Hive Components.....	63
<i>Memory and Hardware Requirements Recommendations</i>	63
<i>Configuring Heap Size and Garbage Collection</i>	64
HiveServer2 Performance Tuning.....	66
<i>Symptoms Displayed When HiveServer2 Heap Memory is Full</i>	66
<i>HiveServer2 Performance Best Practices</i>	67
Tuning Apache Hive on Spark in CDH.....	70
<i>YARN Configuration</i>	71
<i>Spark Configuration</i>	71
<i>Hive Configuration</i>	73
Tuning Apache Hive Performance on the Amazon S3 Filesystem in CDH.....	74
<i>Tuning Hive Write Performance on S3</i>	74
<i>Tuning Hive Table Partition Read Performance on S3</i>	78
<i>Tuning Hive MSCK (Metastore Check) Performance on S3</i>	80
Configuring HMS High Availability in CDH.....	81
<i>Enabling HMS High Availability Using Cloudera Manager</i>	81
Configuring HiveServer2 High Availability in CDH.....	82
<i>Enabling HiveServer2 High Availability Using Cloudera Manager</i>	82
<i>Configuring HiveServer2 to Load Balance Behind a Proxy on Unmanaged Clusters</i>	83
Query Vectorization for Apache Hive in CDH.....	87
<i>Enabling Hive Query Vectorization</i>	87
<i>Tuning Hive Query Vectorization</i>	90
<i>Supported/Unsupported Data Types and Functions</i>	92
<i>Verifying a Query is Vectorized</i>	92

Hive/Impala Replication.....95

Network Latency and Replication.....	95
Host Selection for Hive/Impala Replication.....	95
Hive Tables and DDL Commands.....	96
Replication of Parameters.....	96
Hive Replication in Dynamic Environments.....	96
Guidelines for Snapshot Diff-based Replication.....	97
Replicating from Insecure to Secure Clusters.....	97
Configuring Replication of Hive/Impala Data.....	98
<i>Replication of Impala and Hive User Defined Functions (UDFs)</i>	102
Viewing Replication Schedules.....	102
<i>Enabling, Disabling, or Deleting A Replication Schedule</i>	105

Viewing Replication History.....	105
Hive/Impala Replication To and From Cloud Storage.....	106
Monitoring the Performance of Hive/Impala Replications.....	109
Overview of Apache Hive Security in CDH.....	113
Configuring Encrypted Communication Between HiveServer2 and Client Drivers...114	
Configuring TLS/SSL Encryption for HiveServer2.....	114
<i>Requirements and Assumptions.....</i>	<i>114</i>
<i>Using Cloudera Manager to Enable TLS/SSL.....</i>	<i>114</i>
<i>Client Connections to HiveServer2 Over TLS/SSL.....</i>	<i>115</i>
Configuring SASL Encryption for HiveServer2.....	116
<i>Client Connections to HiveServer2 Using SASL.....</i>	<i>116</i>
Hive SQL Syntax for Use with Sentry.....	117
ALTER DATABASE Statement.....	117
ALTER TABLE Statement.....	117
ALTER VIEW Statement.....	118
CREATE ROLE Statement.....	118
DROP ROLE Statement.....	118
GRANT ROLE Statement.....	118
REVOKE ROLE Statement.....	119
GRANT <Privilege> Statement.....	119
GRANT <Privilege> ON URIs (HDFS and S3A).....	119
REVOKE <Privilege> Statement.....	120
GRANT <Privilege> ... WITH GRANT OPTION.....	120
<i>WITH GRANT OPTION Example.....</i>	<i>121</i>
SET ROLE Statement.....	121
SHOW Statement.....	122
Privileges.....	122
<i>CREATE.....</i>	<i>123</i>
<i>OWNER.....</i>	<i>123</i>
<i>REFRESH (Impala Only).....</i>	<i>124</i>
<i>SELECT.....</i>	<i>124</i>
Troubleshooting Apache Hive in CDH.....	126
Troubleshooting.....	126
<i>HiveServer2 Service Crashes.....</i>	<i>126</i>

Best Practices for Using MSCK REPAIR TABLE.....	127
<i>Example: How MSCK REPAIR TABLE Works.....</i>	<i>128</i>
<i>Guidelines for Using the MSCK REPAIR TABLE Command.....</i>	<i>129</i>
Appendix: Apache License, Version 2.0.....	130

Best Practices for Using Apache Hive in 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:

- [Installation and Upgrade](#)
- [Configuration](#)
- [Using & Managing](#)
- [Tuning](#)
- [Data Replication](#)
- [Security](#)
- [Troubleshooting](#)

Overview of Apache Hive Installation and Upgrade in CDH

Installing:

Hive comes along with the base CDH installation and does not need to be installed manually. Use Cloudera Manager to enable or disable the Hive service. If you disable the Hive service, the component always remains present on the cluster. For details on installing CDH with Cloudera Manager, which installs Hive, see [Installation Using Cloudera Manager Parcels or Packages](#).

Upgrading:

Use Cloudera Manager to upgrade CDH and all of its components, including Hive. For details, see [Upgrading the CDH Cluster](#).

Configuring Apache Hive in CDH

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 for CDH](#) on page 23 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 for CDH](#) on page 10 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 Apache Hive ETL Jobs to Use the Amazon S3 Filesystem in CDH](#) on page 51

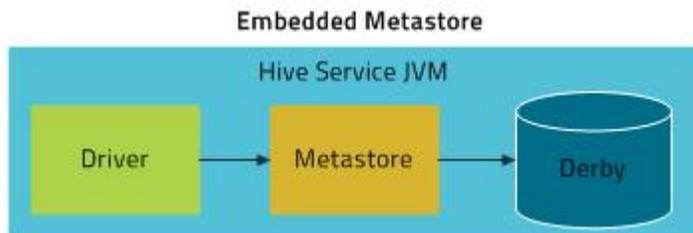
Configuring the Hive Metastore for CDH

The HMS 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

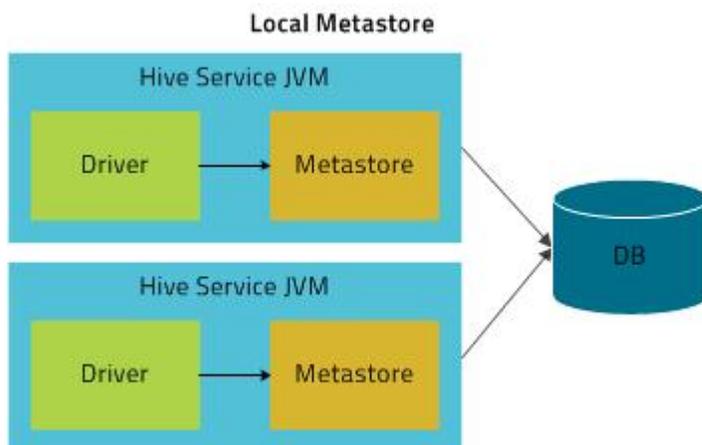
Embedded Mode

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 HiveServer2 process. Both are started for you when you start the HiveServer2 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

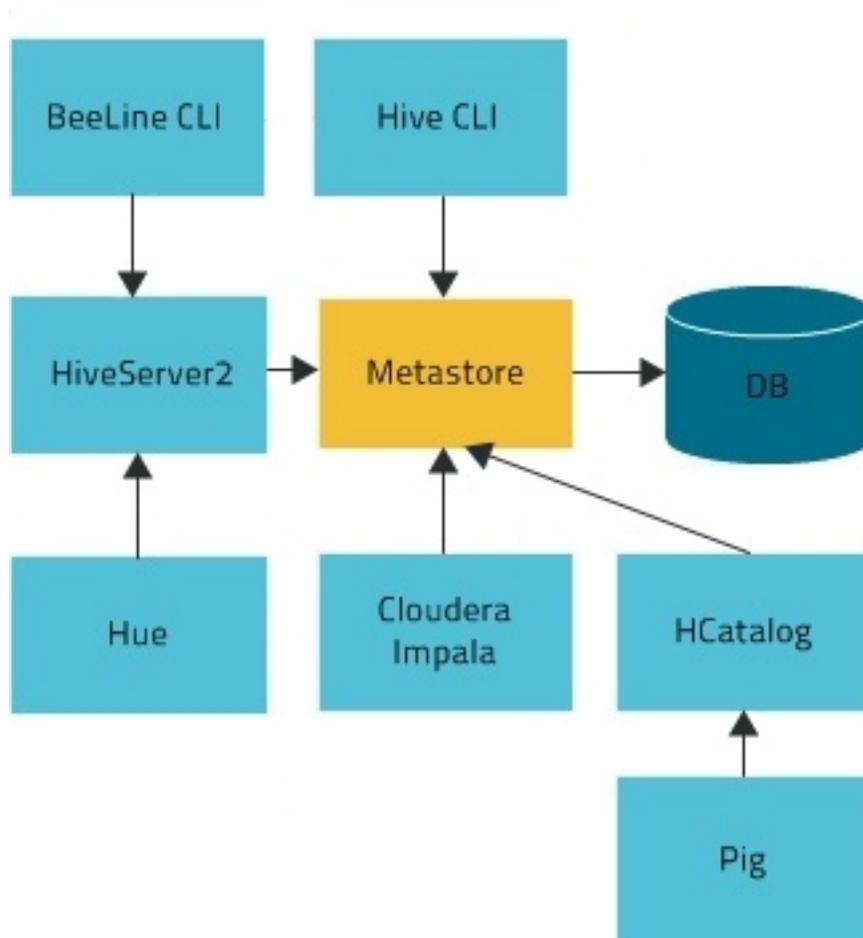


In Local mode, the Hive metastore service runs in the same process as the main HiveServer2 process, but the metastore database runs in a separate process, and can be on a separate host. The embedded metastore service communicates with the metastore database over JDBC.

Remote Mode

Cloudera recommends that you use this mode.

Remote Metastore



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 HiveServer2 process, and the metastore service can all be on the same host, but running the HiveServer2 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 [Database Requirements](#). 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:

- [Install and Configure MariaDB for Cloudera Software](#)

- [Install and Configure MySQL for Cloudera Software](#)
- [Install and Configure PostgreSQL for Cloudera Software](#)
- [Install and Configure Oracle Database for Cloudera Software](#)

Metastore Memory and Hardware Requirements

Component	Java Heap		CPU	Disk
HiveServer 2	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required for the following: <ul style="list-style-type: none"> • HiveServer2 log files • <code>stdout</code> and <code>stderr</code> output files • Configuration files • Operation logs stored in the <code>operation_logs_dir</code> directory, which is configurable • Any temporary files that might be created by local map tasks under the <code>/tmp</code> directory
	2-10 connections	4-6 GB		
	11-20 connections	6-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Cloudera recommends splitting HiveServer2 into multiple instances and load balancing them once you start allocating more than 16 GB to HiveServer2. The objective is to adjust the size to reduce the impact of Java garbage collection on active processing by the service.			
Set this value using the Java Heap Size of HiveServer2 in Bytes Hive configuration property. For more information, see Tuning Hive in CDH .				
Hive Metastore	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required so that the Hive metastore can store the following artifacts: <ul style="list-style-type: none"> • Logs • Configuration files • Backend database that is used to store metadata if the database server is also hosted on the same node
	2-10 connections	4-10 GB		
	11-20 connections	10-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Set this value using the Java Heap Size of Hive Metastore Server in Bytes Hive configuration property. For more information, see Tuning Hive in CDH .			
Beeline CLI	Minimum: 2 GB		N/A	N/A



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 the Hive metastore, as well as HiveServer2 and Hive clients, see [Tuning Apache Hive in CDH](#) on page 63.

General Metastore Tuning Recommendations

Generally, you need to limit concurrent connections to Hive metastore. A large number of open connections affects performance as does issues with the backend database, improper Hive use, such as extremely complex queries, a connection leak, and other issues. Try making the following changes:

- Buy an SSD for one or more Hive metastores.
- Cloudera recommends that a single query access no more than 10,000 table partitions. If the query joins tables, calculate the combined partition count accessed across all tables.
- Tune the backend (the RDBMS). HiveServer connects to HMS, and only HMS connects to the RDBMS. The longer the backend takes, the more memory the HMS needs to respond to the same requests. Limit the number of connections in the backend database.

MySQL: For example, in `/etc/my.cnf`:

```
[mysqld]
datadir=/var/lib/mysql
max_connections=8192
. . .
```

MariaDB: For example, in `/etc/systemd/system/mariadb.service.d/limits.conf`:

```
[Service]
LimitNOFILE=24000
. . .
```

- Use default thrift properties (8K):

```
hive.server2.async.exec.threads 8192
hive.server2.async.exec.wait.queue.size 8192
hive.server2.thrift.max.worker.threads 8192
```

- Set `datanucleus.connectionPool.maxPoolSize` for your applications. For example, if `poolSize = 100`, with 3 HMS instances (one dedicated to compaction), and with 4 pools per server, you can accommodate 1200 connections.

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](#).

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 reside on a host or hosts separate from the HiveServer2 process. MySQL is the most popular database to use. Use the following

steps to configure a remote metastore. If you are planning to use a cloud service database, such as Amazon Relational Database Service (RDS), see [Configuring a Shared Amazon RDS as an HMS for CDH](#) on page 54 for information about how to set up a shared Amazon RDS as your Hive metastore.

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 JDBC driver

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

For instructions on installing the MySQL JDBC driver, see [Installing the MySQL JDBC Driver](#).

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 [Metastore 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, HiveServer2), `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.conf` 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 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, HiveServer2), `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 (or `ojdbc8.jar`, for Oracle Database 12.2).



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, HiveServer2), `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>
```

Specifying a JDBC URL Override for Database Connections

In instances where you wish to configure fine-grained tuning of the HMS database connection, you can specify a JDBC URL override to be used when establishing a connection to the HMS database.



Warning:

This configuration setting is intended for advanced database users only. Be aware that when using this override, the following properties are overwritten (in other words, their values will *not* be used):

- Hive Metastore Database Name
- Hive Metastore Database Host
- Hive Metastore Database Port
- Enable TLS/SSL to the Hive Metastore Database

Prerequisites

- The required default user role is [Configurator](#).
- When using the **Hive Metastore Database JDBC URL Override**, you *must* still provide the following properties to connect to the database:
 - Hive Metastore Database Type
 - Hive Metastore Database User
 - Hive Metastore Database Password

To specify a Hive Metastore JDBC URL Override for database connections:

1. Open the Cloudera Manager Admin Console and go to the **Hive-1 service**.
2. Click the **Configuration** tab.
3. Select **Category > Hive Metastore Database**.
4. Edit the **Hive Metastore Database JDBC URL Override** property according to your cluster configuration (the default value is Empty " "):

Database Type	Hive Metastore Database JDBC URL Override Format
MySQL	<code>jdbc:mysql://<host>:<port>/<metastore_db>?key=value</code>
PostgreSQL	<code>jdbc:postgresql://<host>:<port>/<metastore_db>?key=value</code>
Oracle JDBC Thin using a Service Name	<code>jdbc:oracle:thin:@//<host>:<port>/<service_name></code>
Oracle JDBC Thin using SID	<code>jdbc:oracle:thin:@<host>:<port>:<SID></code>
Oracle JDBC Thin using TNSName	<code>jdbc:oracle:thin:@<TNSName></code>

Important: Formats are dependent on the JDBC driver version that you are using and subject to change between releases. Refer to your database product documentation to confirm JDBC formats for the specific database version you are using.

Configuring HiveServer2 for CDH

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

Important: Because of concurrency and security issues, HiveServer1 was deprecated in CDH 5.3 and has been removed from CDH 6.

HiveServer2 Memory and Hardware Requirements

Component	Java Heap		CPU	Disk
HiveServer 2	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required for the following: <ul style="list-style-type: none"> • HiveServer2 log files • <code>stdout</code> and <code>stderr</code> output files • Configuration files • Operation logs stored in the <code>operation_logs_dir</code> directory, which is configurable • Any temporary files that might
	2-10 connections	4-6 GB		
	11-20 connections	6-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Cloudera recommends splitting HiveServer2 into multiple instances and load balancing them once you start allocating more than 16 GB to HiveServer2. The objective is to adjust the size to reduce the impact of Java garbage			

Component	Java Heap		CPU	Disk
	collection on active processing by the service. Set this value using the Java Heap Size of HiveServer2 in Bytes Hive configuration property. For more information, see Tuning Hive in CDH .			be created by local map tasks under the /tmp directory
Hive Metastore	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required so that the Hive metastore can store the following artifacts: <ul style="list-style-type: none"> • Logs • Configuration files • Backend database that is used to store metadata if the database server is also hosted on the same node
	2-10 connections	4-10 GB		
	11-20 connections	10-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	Set this value using the Java Heap Size of Hive Metastore Server in Bytes Hive configuration property. For more information, see Tuning Hive in CDH .			
Beeline CLI	Minimum: 2 GB		N/A	N/A

 **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 [Tuning Apache Hive in CDH](#) on page 63 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

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 for HiveServer2:

Connection URL	Driver Class
<code>jdbc:hive2://<host>:<port></code>	<code>org.apache.hive.jdbc.HiveDriver</code>

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](#), which allows you to use a custom authentication provider for HiveServer2; and [HiveServer2 Impersonation](#), 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



Important: Because of concurrency and security issues, HiveServer1 was deprecated in CDH 5.3 and has been removed from CDH 6. The Hive CLI is deprecated 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 binds to port 10000 by default. Set the port for HiveServer2 in the `hive.server2.thrift.port` property in the `hive-site.xml` file. 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 for HiveServer2 by setting these environment variables:

Port	Host Address
<code>HIVE_SERVER2_THRIFT_PORT</code>	<code>HIVE_SERVER2_THRIFT_BIND_HOST</code>

Starting the Hive Metastore in CDH

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
```

Apache Hive File System Permissions in CDH

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](#) 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 in CDH

HiveServer2 is an improved version of HiveServer that supports Kerberos authentication and multi-client concurrency. You can access HiveServer2 by using the Beeline client.



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.

Note that because of concurrency and security issues, HiveServer1 was deprecated in CDH 5.3 and has been removed from CDH 6.

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](#).

There are 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](#).

Using Apache Hive with HBase in CDH

You need to associate the HBase service with the Hive service, so Hive scripts can use HBase as described in the procedure below.

Prerequisites for Using Hive with HBase

- Using Cloudera Manager, [install CDH](#) with HBase using Cloudera Manager. Add the Hive service, and set up the cluster.
- If Kerberos is enabled, assign the following roles to the gateway node:
 - Hive Gateway Role
 - HDFS Gateway Role
 - HBase Gateway Role
- If Kerberos is not enabled, assign the following roles to the gateway node:

Configuring Apache Hive in CDH

- Hive Gateway Role
- HDFS Gateway Role

Associate HBase with Hive

1. In a Sentry-controlled environment, grant the `hive` user most privileges in HBase.

```
grant 'hive', 'RWXC'
```

2. From the Cloudera Manager home page, click the Hive service.
3. On the Hive service page, select the **Configuration** tab.
4. On the Hive service Configuration page, type `hbase` into the search text box.
5. Locate the **HBase Service** configuration property on the page, select the HBase instance that you want to associate with Hive, and click **Save Changes**.
6. Redeploy the client configuration for the Hive service and restart all stale services.

The HBase service is now associated with the Hive service, and your Hive scripts can use HBase.

Using the Metastore Schema Tool in CDH

Use the Metastore command-line `schematool` to upgrade or validate the metastore database schema for unmanaged clusters.



Note:

If you are using Cloudera Manager to manage your clusters, the Metastore `schematool` is also available in the Hive service page to validate or upgrade the metastore:

1. From the Cloudera Manager Admin console, select the Hive service.
2.
 - To validate the schema, on the Hive service page, click **Actions**, and select **Validate Hive Metastore Schema**.
 - To upgrade the schema:
 1. On the Hive service page, click **Actions**, and select **Stop** to stop the service.
 2. Still on the Hive service page, click **Actions**, and select **Upgrade Hive Database Metastore Schema**.
 3. After the upgrade completes, restart the service.

Schema Version Verification and Validation

Hive 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 configuration properties that implicitly create or alter the existing schema are disabled by default. Consequently, Hive does not attempt to change the metastore schema implicitly. When you execute a Hive query against a metastore where the schema is not initialized or the schema is old, it fails to access the metastore and an entry similar to the following example appears in the error log:

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

Use Hive `schematool` to repair the condition that causes this error by either initializing the schema or upgrading it.

Using schematool

Use the Metastore `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 metastore 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
  -validate                     Validate the database
  -verbose                      Only print SQL statements
```

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

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

Prerequisite Configuration

Before you can use the `schematool`, you must add the following properties to the `/etc/hive/conf/hive-site.xml` file:

- `javax.jdo.option.ConnectionURL`
- `javax.jdo.option.ConnectionDriverName`

For example, the following `hive-site.xml` entries are made if you are using a MySQL database as your Hive metastore and `hive1` is the database user name:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://my_cluster.com:3306/hive1?useUnicode=true&amp;characterEncoding=UTF-8</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
```



Note: The Hive Schema tool does not support using TLS/SSL encryption to HMS database.

Usage Examples

To use the `schematool` command-line tool, navigate to the directory where it is located:

- If you installed CDH using parcels, schematool is usually located at:

```
/opt/cloudera/parcels/CDH/lib/hive/bin/schematool
```

- If you installed CDH using packages, schematool is usually located at:

```
/usr/lib/hive/bin/schematool
```

After you locate the executable, you can use schematool to perform the following actions:

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

```
$ schematool -dbType mysql -initSchema -passWord <db_user_pswd> -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 metastore schema initialization to <new_version>
Initialization script hive-schema-<new_version_number>.mysql.sql
Initialization script completed
schemaTool completed
```

- Get schema information using the `info` option.

```
$ schematool -dbType mysql -info -passWord <db_user_pswd> -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>
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 or if the schema is not initialized, the tool reports an error as follows.

```
$ schematool -dbType mysql -info -passWord <db_user_pswd> -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>
Hive distribution version:      <new_version>
Required schema version:      <new_version>
org.apache.hadoop.hive.metastore.HiveMetaException: Failed to get schema version,
Cause:<cause_description>
*** schemaTool failed ***
```

- You can upgrade the schema from a specific release by specifying the `-upgradeSchemaFrom` option. The `-upgradeSchemaFrom` option requires the Hive version and not the CDH version. See [CDH 6 Packaging Information](#) for information about which Hive version ships with each CDH release. The following example shows how to upgrade from CDH 5.2/Hive 0.13.1:

```
$ 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
```

- Use the `-validate` option to verify the metastore schema. The following example shows the types of validations that are performed against the metastore schema when you use this option with `schematool`:

```
$ schematool -dbType mysql -passWord <db_user_pswd> -userName
  <db_user_name> -validate
Starting metastore validation

Validating schema version
Succeeded in schema version validation.
[SUCCESS]

Validating sequence number for SEQUENCE_TABLE
Succeeded in sequence number validation for SEQUENCE_TABLE
[SUCCESS]

Validating metastore schema tables
Succeeded in schema table validation.
[SUCCESS]

Validating database/table/partition locations
Succeeded in database/table/partition location validation
[SUCCESS]

Validating columns for incorrect NULL values
Succeeded in column validation for incorrect NULL values
[SUCCESS]

Done with metastore validation: [SUCCESS]
schemaTool completed
```

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

```
$ schematool -dbType mysql -upgradeSchemaFrom 0.10.0 -dryRun -passWord
  <db_user_pswd> -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 0.10.0 to <new_version>
Upgrade script upgrade-0.10.0-to-0.11.0.mysql.sql
Upgrade script upgrade-0.11.0-to-0.12.0.mysql.sql
Upgrade script upgrade-0.12.0-to-0.13.0.mysql.sql
Upgrade script upgrade-0.13.0-to-0.14.0.mysql.sql
Upgrade script upgrade-0.14.0-to-1.1.0.mysql.sql
Upgrade script upgrade-1.1.0-to-<new_version>.mysql.sql
schemaTool completed
```

Installing Cloudera JDBC and ODBC Drivers on Clients in CDH

To install Cloudera Hive JDBC drivers or ODBC drivers, go to the following download pages on Cloudera's web site.

Cloudera Hive JDBC Driver Download

The Cloudera Hive JDBC Driver versions 2.5.20, 2.6.1, and later have been tested with CDH 6.0. Cloudera recommends that you use these versions with Hive when you upgrade to CDH 6.0.

To download the Cloudera Hive JDBC Driver, go to:

Configuring Apache Hive in CDH

www.cloudera.com/downloads/connectors/hive/jdbc/2-5-20.html.

Choose version 2.5.20 or later from the **Version** drop-down list on the page.

For documentation about installing and using the driver, see the guide for the appropriate driver type and version at the [Cloudera Enterprise Connector Documentation page](#).

Cloudera Hive ODBC Driver Download

The Cloudera Hive ODBC driver version 2.5.25 and later has been tested with CDH 6.0. Cloudera recommends that you use these versions when you upgrade to CDH 6.0.

To download the Cloudera Hive ODBC driver, go to:

www.cloudera.com/downloads/connectors/hive/odbc/2-5-25.html.

Choose version 2.5.25 or later from the **Version** drop-down list on the page.

For documentation about installing and using the driver, see the guide for the appropriate driver type and version at the [Cloudera Enterprise Connector Documentation page](#).

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 Hive Metastore to Use HDFS High Availability in CDH

In this task, you configure a host for connecting to the backend database. You can either configure environment variables or add configuration properties to `hive-site.xml`. Configure a limited number of hosts to limit exposing the backend Hive database username, password, and connection string. To configure environment variables and run the metatool, follow this procedure:

1. Connect to any Hive Gateway host, Hive Metastore (HMS), or HiveServer (HS2) host.
2. Set `HIVE_CONF_DIR` to the Hive MetaStore process directory.
3. Set `HADOOP_CREDSTORE_PASSWORD` to the same value defined in the `supervisor.conf` file (which is located in the Hive MetaStore process directory)
4. Set `AUX_CLASSPATH` to the value of your your Hive MetaStore process directory, which is defined in `logs/stderr.log`.

For example:

```
export HIVE_CONF_DIR=/var/run/cloudera-scm-agent/process/4595-hive-HIVEMETASTORE
export HADOOP_CREDSTORE_PASSWORD=abcdefghijkl234...
export AUX_CLASSPATH=/opt/cloudera/parcels/CDH-5.13.3-1.cdh5.13.3.p0.2....
```

5. Run the following command to connect to the database and list FS roots:

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

The output is:

```
Listing FS Roots..
hdfs://[hostname]:8020/user/hive/warehouse
```

Alternatively, instead of setting and exporting environment variables, open the `hive-site.xml` file in `/etc/hive/conf/`. Add the following properties to the `hive-site.xml`:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://[ENTER BACKEND DATABASE HOSTNAME]:[ENTER PORT]/[ENTER HIVE BACKEND
  DATABASE USERNAME]?useUnicode=true&characterEncoding=UTF-8</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>[ENTER BACKEND DATABASE USERNAME]</value>
</property>
<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>[ENTER BACKEND DATABASE PASSWORD]</value>
</property>
```

To determine what the backend database host, port, username, or password is, in Cloudera Manager, go to **Hive > Configurations**. Set **Category Filter** to `Hive Metastore Database`. The password is not exposed in plaintext.

To configure other CDH components to use HDFS high availability, see [Configuring Other CDH Components to Use HDFS HA](#).

Using & Managing Apache Hive in CDH

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. 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](#).

For information about configuring a remote Hive metastore database with Cloudera Manager, see [Step 4: Install and Configure Databases](#). To configure high availability for the Hive metastore, see [Configuring HMS High Availability in CDH](#) on page 81.

- **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](#) 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. Enter a **Reason for change**, and then click **Save Changes** to commit the changes.
 6. Return to the Home page by clicking the Cloudera Manager logo.

7. Click the  icon that is next to any stale services to invoke the cluster restart wizard.
8. Click **Restart Stale Services**.
9. Click **Restart Now**.
10. Click **Finish**.

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.

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.

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. Enter a **Reason for change**, and then 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.

Overview of Ingesting and Querying Data with Apache Hive in CDH

Data ingestion begins your data pipeline or "write path." Classic data pipelines bring in data and then apply ETL operations on it, which clean and transform data for consumption. Apache Hive in CDH is the preferred tool for ETL workloads. Hive queries transform data, using functions such as CAST or TRIM, or by joining data sets, to ensure the data conforms to target data models for your data warehouse. Then the data can be consumed. For example, by business intelligence (BI) tools or users' ad-hoc queries.

Ingesting Data with Hive

Hive can ingest data in 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. See [Using Apache Parquet Data Files with CDH](#) for general information about the Parquet file format and for information about using Parquet tables in Hive. 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.

See [Using Avro Data Files in Hive](#) for details about using Avro to ingest data into Hive tables and about using Snappy compression on the output files.

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 [Accessing Apache Hive Table Statistics in CDH](#) on page 47 for details about collecting statistics for Hive.

Transaction (ACID) Support in Hive

The CDH distribution of Hive 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.

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](#).

Apache Parquet Tables with Hive in CDH

[Apache Parquet](#) is a [columnar storage](#) format available to any component in the Hadoop ecosystem, regardless of the data processing framework, data model, or programming language. The Parquet file format incorporates several features that support data warehouse-style operations:

- Columnar storage layout - A query can examine and perform calculations on all values for a column while reading only a small fraction of the data from a data file or table.
- Flexible compression options - Data can be compressed with any of several codecs. Different data files can be compressed differently.
- Innovative encoding schemes - Sequences of identical, similar, or related data values can be represented in ways that save disk space and memory. The encoding schemes provide an extra level of space savings beyond overall compression for each data file.
- Large file size - The layout of Parquet data files is optimized for queries that process large volumes of data, with individual files in the multi-megabyte or even gigabyte range.

Parquet is automatically installed when you install CDH, and the required libraries are automatically placed in the classpath for all CDH components. Copies of the libraries are in `/usr/lib/parquet` or `/opt/cloudera/parcels/CDH/lib/parquet`.

CDH lets you use the component of your choice with the Parquet file format for each phase of data processing. For example, you can read and write Parquet files using Pig and MapReduce jobs. You can convert, transform, and query Parquet tables through Hive, Impala, and Spark. And you can interchange data files between all of these components.

Using Parquet Tables in Hive

To create a table named `PARQUET_TABLE` that uses the Parquet format, use a command like the following, substituting your own table name, column names, and data types:

```
CREATE TABLE parquet_table_name (x INT, y STRING) STORED AS PARQUET;
```



Note:

- Once you create a Parquet table, you can query it or insert into it through other components such as Impala and Spark.
- Set `dfs.block.size` to 256 MB in `hdfs-site.xml`.
- To enhance performance on Parquet tables in Hive, see [Enabling Query Vectorization](#).

If the table will be populated with data files generated outside of Impala and Hive, you can create the table as an external table pointing to the location where the files will be created:

```
CREATE EXTERNAL TABLE parquet_table_name (x INT, y STRING)
LOCATION '/test-warehouse/tinytable'
STORED AS PARQUET;
```

To populate the table with an `INSERT` statement, and to read the table with a `SELECT` statement, see [Loading Data into Parquet Tables](#).

To set the compression type to use when writing data, configure the `parquet.compression` property:

```
SET parquet.compression=GZIP;
INSERT OVERWRITE TABLE tinytable SELECT * FROM texttable;
```

The supported compression types are `UNCOMPRESSED`, `GZIP`, and `SNAPPY`.

Running Apache Hive on Spark in CDH

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 34.
- 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 38.

Hive on Spark Memory and Hardware Requirements

Component	Memory	CPU	Disk
Hive-on-Spark	<ul style="list-style-type: none"> • Minimum: 16 GB • Recommended: 32 GB for larger data sizes <p>Individual executor heaps should be no larger than 16 GB so machines with more RAM can use multiple executors.</p>	<ul style="list-style-type: none"> • Minimum: 4 cores • Recommended: 8 cores for larger data sizes 	Disk space requirements are driven by scratch space requirements for Spark spill.
For more information on how to reserve YARN cores and memory that will be used by Spark executors, refer to Tuning Apache Hive on Spark in CDH on page 70.			

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. In the Cloudera Manager Admin Console, 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**.
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 the  icon that is next to any stale services 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 Apache Hive on Spark in CDH](#) on page 70.

Dynamic Partition Pruning for Hive Map Joins

You can enable dynamic partition pruning for map joins when you are running Hive on Spark (HoS). Dynamic partition pruning (DPP) is a database optimization that can significantly decrease the amount of data that a query scans, thereby executing your workloads faster. DPP achieves this by dynamically determining and eliminating the number of partitions that a query must read from a partitioned table.

Map joins also optimize how Hive executes queries. They cause a small table to be scanned and loaded in memory as a hash table so that a fast join can be performed entirely within a mapper without having to use another reduce step. If you have queries that join small tables, map joins can make them execute much faster. Map joins are enabled by default in CDH with the **Enable MapJoin Optimization** setting for HiveServer2 in Cloudera Manager. Hive automatically uses map joins for join queries that involve a set of tables where:

- There is one large table and there is no limit on the size of that large table.
- All other tables involved in the join must have an aggregate size under the value set for **Hive Auto Convert Join Noconditional Size** for HiveServer2, which is set to 20MB by default in Cloudera Manager.

For more information about map joins, see the [Apache wiki](#).

To enable or disable map joins on a per-query basis, use the Hive `SET` command:

```
SET hive.auto.convert.join=true;
SET hive.auto.convert.join.noconditionaltask.size=<number_in_megabytes>;
```

When you are using HoS and the tables involved in a join query trigger a map join, two Spark jobs are launched and perform the following actions:

- the first job scans the smaller table, creates a hash table, and writes it to HDFS,
- the second job runs the join and the rest of the query, scanning the larger table.

If DPP is enabled and is also triggered, the two Spark jobs perform the following actions:

- the first Spark job creates the hash table from the small table and identifies the partitions that should be scanned from the large table,
- the second Spark job then scans the relevant partitions from the large table that are to be used in the join.

After these actions are performed, the query proceeds normally with the map join.

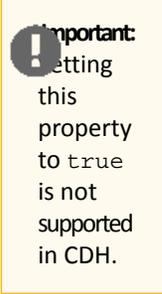
Enabling Dynamic Partition Pruning for Map Joins in Hive on Spark

Dynamic partition pruning (DPP) is disabled by default. Use Cloudera Manager to set the following properties.



Important: Cloudera does not support nor recommend setting the property `hive.spark.dynamic.partition.pruning` to `true` in production environments. This property enables DPP for all joins, both map joins and common joins. The property `hive.spark.dynamic.partition.pruning.map.only`, which enables DPP for map joins only in Hive on Spark is the only supported implementation of DPP for Hive on Spark in CDH.

Property Name	Description	Default Setting
<code>hive.spark.dynamic.partition.pruning.map.only</code>	Enables dynamic partition pruning for queries where the join on the partitioned column is a map join. This property only applies to the Spark execution engine. Set this property to <code>true</code> to use dynamic partition pruning for queries where the join on the partitioned column is a map join.	false (turned off)
<code>hive.spark.dynamic.partition.pruning</code>	Enables dynamic partition pruning for <i>all</i> joins, including shuffle joins and map joins.	false (turned off)

Property Name	Description	Default Setting
		

Enabling DPP on a Per-Query Basis with the Hive SET Command

To enable DPP at the session level, use the Hive `SET` command:

```
SET hive.spark.dynamic.partition.pruning.map.join.only=true;
```

Enabling DPP as a Service-Wide Default with Cloudera Manager

Use Cloudera Manager to enable DPP as a service-wide default:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope and click the **Performance** category.
4. Search for **Hive on Spark Dynamic Partition Pruning for MapJoins**, and select the check box.
5. Click **Save Changes**.

Verifying Your Query Uses Dynamic Partition Pruning in Hive on Spark

Use `EXPLAIN` to generate a query plan, which you can use to verify that DPP is being triggered for your query.

Example of Verifying that Dynamic Partition Pruning Is Triggered For Your Query

In this example, TPC-DS benchmark data is used with the query generated from `query3.tpl` in their downloadable package. It demonstrates how you can use the `EXPLAIN` command to verify that DPP is being triggered. For more information about the TPC-DS benchmark data and queries, see www.tpc.org/tpcds/.

First, set the following properties which instruct Hive to use Spark as its execution engine and turns on DPP for map joins:

```
SET hive.execution.engine=spark;
SET hive.spark.dynamic.partition.pruning.map.join.only=true;
```

Then run the following commands, which tell Hive to use the `testing_example_db` database and to show (`EXPLAIN`) the query plan for the query that follows:

```
USE testing_example_db;

EXPLAIN
SELECT dt.d_year
       ,item.i_brand_id brand_id
       ,item.i_brand brand
       ,sum(ss_ext_sales_price) sum_agg
FROM date_dim dt
     ,store_sales
     ,item
WHERE dt.d_date_sk = store_sales.ss_sold_date_sk
```

```

AND store_sales.ss_item_sk = item.i_item_sk
AND item.i_manufact_id = 436
AND dt.d_moy=12
GROUP BY dt.d_year
      ,item.i_brand
      ,item.i_brand_id
ORDER BY dt.d_year
      ,sum_agg desc
      ,brand_id
LIMIT 100;

```

The `EXPLAIN` command returns the query plan for the TPC-DS query. An excerpt from that query plan is included below. Look for the **Spark HashTable Sink Operator** and the **Spark Partition Pruning Sink Operator**, which are in bold font in the following output. Presence of these sink operators in the query plan indicate that DPP is being triggered for the query.

```

+-----+
| Explain |
+-----+
STAGE DEPENDENCIES:
  Stage-2 is a root stage
  Stage-1 depends on stages: Stage-2
  Stage-0 depends on stages: Stage-1

STAGE PLANS:
  Stage: Stage-2
    Spark
      DagName: hive_20170908151313_f478b7d3-89b8-4c6d-b98c-4ef3b8e25bf7:964 |
      Vertices:
        Map 1
          Map Operator Tree:
            TableScan
              alias: dt
              filterExpr: (d_date_sk is not null and (d_moy = 12)) (type: boolean)
              Statistics: Num rows: 73049 Data size: 2045372 Basic stats: COMPLETE
            Column stats: NONE |
            Filter Operator
              predicate: (d_date_sk is not null and (d_moy = 12)) (type: boolean)
              Statistics: Num rows: 18262 Data size: 511336 Basic stats: COMPLETE
            Column stats: NONE |
            Spark HashTable Sink Operator
              keys:
                0 d_date_sk (type: bigint)
                1 ss_sold_date_sk (type: bigint) |
            Select Operator
              expressions: d_date_sk (type: bigint) |
              outputColumnNames: _col0
              Statistics: Num rows: 18262 Data size: 511336 Basic stats:
            COMPLETE Column stats: NONE |
            Group By Operator
              keys: _col0 (type: bigint)
              mode: hash
              outputColumnNames: _col0
              Statistics: Num rows: 18262 Data size: 511336 Basic stats:
            COMPLETE Column stats: NONE |
            Spark Partition Pruning Sink Operator
              partition key expr: ss_sold_date_sk |
              tmp Path:
https://server-name.com:8020/tmp/hive/hive/8934148311-40611055f9c328/hive\_2017-09-08\_15-13-54\_861\_527212517384722-4/-mr-10003/2/1
              Statistics: Num rows: 18262 Data size: 511336 Basic stats:
            COMPLETE Column stats: NONE |
              target column name: ss_sold_date_sk |
              target work: Map 2
            Local Work:
              Map Reduce Local Work
            Map 5

```

```

Map Operator Tree:
  TableScan
    alias: item
    filterExpr: (i_item_sk is not null and (i_manufact_id = 436)) (type:
boolean) |
  Statistics: Num rows: 102000 Data size: 2244000 Basic stats: COMPLETE
Column stats: NONE |
  Filter Operator
    predicate: (i_item_sk is not null and (i_manufact_id = 436)) (type:
boolean) |
  Statistics: Num rows: 25500 Data size: 561000 Basic stats: COMPLETE
Column stats: NONE |
  Spark HashTable Sink Operator
    keys:
      0 _col32 (type: bigint)
      1 i_item_sk (type: bigint)
Local Work:
  Map Reduce Local Work
...

```



Note: There are a few map join patterns that are not supported by DPP. For DPP to be triggered, the **Spark Partition Pruning Sink Operator** must have a target Map Work in a child stage. For example, in the above query plan, the **Spark Partition Pruning Sink Operator** resides in **Stage-2** and has a **target work: Map 2**. So for DPP to be triggered, **Map 2** must reside in either **Stage 1** or **Stage 0** because both are dependent on **Stage 2**, thus they are both children of **Stage 2**. See the **STAGE DEPENDENCIES** at the top of the query plan to see the stage hierarchy. If **Map 2** resides in **Stage 2**, DPP is not triggered because **Stage 2** is the root stage and therefore cannot be a child stage.

Queries That Trigger and Benefit from Dynamic Partition Pruning in Hive on Spark

When tables are created in Hive, it is common practice to partition them. Partitioning breaks large tables into horizontal slices of data. Each partition typically corresponds to a separate folder on HDFS. Tables can be partitioned when the data has a "natural" partitioning column, such as a date column. Hive queries that read from partitioned tables typically filter on the partition column in order to avoid reading all partitions from the table. For example, if you have a partitioned table called `date_partitioned_table` that is partitioned on the `datetime` column, the following query only reads partitions that are created after January 1, 2017:

```

SELECT *
FROM date_partitioned_table
WHERE datetime > '2017-01-01';

```

If the `date_partitioned_table` table has partitions for dates that extend to 2010, this `WHERE` clause filter can significantly decrease the amount of data that needs to be read by the query. This query is easy for Hive to optimize. When it is compiled, only partitions where `datetime` is greater than `2017-01-01` need to be read. This form of partition pruning is known as *static partition pruning*.

However, when queries become more complex, the filter on the partitioned column cannot be evaluated at runtime. For example, this query:

```

SELECT *
FROM date_partitioned_table
WHERE datetime IN (SELECT * FROM non_partitioned_table);

```

With this type of query, it is difficult for the Hive compiler to optimize its execution because the rows that are returned by the sub query `SELECT * FROM non_partitioned_table` are unknown. In this situation, dynamic partition pruning (DPP) optimizes the query. Hive can dynamically prune partitions from the scan of `non_partitioned_table`

by eliminating partitions while the query is running. Queries that use this pattern can see performance improvements when DPP is enabled. Note that this query contains an `IN` clause which triggers a join between the `date_partitioned_table` and the `non_partitioned_table`. DPP is only triggered when there is a join on a partitioned column.

DPP might provide performance benefits for Hive data warehouses that use the star or snowflake schema. Performance improvements are possible for Hive queries that join a partitioned fact table on the partitioned column of a dimension table if DPP is enabled. The TPC-DS benchmark is a good example where many of its queries benefit from DPP. The query example from the TPC-DS benchmark listed in the [above section with EXPLAIN](#), triggers DPP:

```
SELECT dt.d_year
      ,item.i_brand_id brand_id
      ,item.i_brand brand
      ,sum(ss_ext_sales_price) sum_agg
FROM date_dim dt
     ,store_sales
     ,item
WHERE dt.d_date_sk = store_sales.ss_sold_date_sk
     AND store_sales.ss_item_sk = item.i_item_sk
     AND item.i_manufact_id = 436
     AND dt.d_moy=12
GROUP BY dt.d_year
        ,item.i_brand
        ,item.i_brand_id
ORDER BY dt.d_year
        ,sum_agg desc
        ,brand_id
LIMIT 100;
```

This query performs a join between the partitioned `store_sales` table and the non-partitioned `date_dim` table. The join is performed against the partition column for `store_sales`, which is what triggers DPP. The join must be against a partitioned column for DPP to be triggered.

DPP is only supported for map joins. It is not supported for common joins, those that require a shuffle phase. A single query may have multiple joins, some of which are map joins and some of which are common joins. Only the join on the partitioned column must be a map join for DPP to be triggered.

Debugging Dynamic Partition Pruning in Hive on Spark

Debug DPP for Hive on Spark by viewing the query plan produced with the `EXPLAIN` command or by viewing two types of log files. Both options are discussed in the following sections.

Debugging with Query Plans Produced with `EXPLAIN`

A simple way to check whether DPP is triggered for a query is to use the `EXPLAIN` command as shown above in [Verifying Your DPP Configuration in Hive on Spark](#). If the query plan contains a `Spark Partition Pruning Sink Operator`, DPP will be triggered for the query. If it does not contain this operator, DPP will not be triggered for the query.

Debugging with Logs

Use the `HiveServer2` logs to debug the compile time phase of DPP and use the `Hive on Spark Remote Driver` logs to debug the runtime phase of DPP:

- **HiveServer2 Logs**

The `HiveServer2` logs print debugging information from the Java class `DynamicPartitionPruningOptimization`. This class looks at the query and checks if it can benefit from DPP. If the query can benefit from DPP, the class modifies the query plan to include DPP-specific operators, such as the `Spark Partition Pruning Sink Operator`. When the class runs, it prints out information related to whether or not it is enabling DPP for a particular clause in the query.

Using & Managing Apache Hive in CDH

For example, if the following message appears in the HiveServer2 log, it means that DPP will be triggered and that partitions will be dynamically pruned from the `partitioned_table` table, which is in bold text in the following example:

```
INFO org.apache.hadoop.hive.q1.optimizer.DynamicPartitionPruningOptimization:
[HiveServer2-Handler-Pool: Thread-xx]: Dynamic partitioning:
default@partitioned_table.partition_column
```

To access these log files in Cloudera Manager, select **Hive > HiveServer2 > Log Files > Role Log File**.

- **Hive on Spark Remote Driver Logs**

The Hive on Spark (HoS) Remote Driver logs print debugging information from the Java class `SparkDynamicPartitionPruner`. This class does the actual pruning of the partitioned table. Because pruning happens at runtime, the logs for this class are located in the HoS Remote Driver logs instead of the HiveServer2 logs. These logs print which partitions are pruned from the partitioned table, which can be very useful for troubleshooting.

For example, if the following message appears in the HoS Remote Driver log, it means that the partition `partition_column=1` is being pruned from the table `partitioned_table`, both of which are in bold text in the following example:

```
INFO spark.SparkDynamicPartitionPruner:Pruning path:
hdfs://<namenode_uri>/user/hive/warehouse/partitioned_table/partition_column=1
```

To access these log files in Cloudera Manager, select **SPARK_ON_YARN > History Server Web UI > <select_an_application> > Executors > executor id = driver > stderr**.

Using Hive UDFs with Hive on Spark

When the execution engine is set to Spark, use Hive UDFs the same way that you use them when the execution engine is set to MapReduce. To apply a custom UDF on the column of a Hive table, use the following syntax:

```
SELECT <custom_UDF_name>(<column_name>) FROM <table_name>;
```

For example, to apply the custom UDF `addfunc10` to the `salary` column of the `sample_07` table in the default database that ships with CDH, use the following syntax:

```
SELECT addfunc10(salary) FROM sample_07 LIMIT 10;
```

The above HiveQL statement returns only 10 rows from the `sample_07` table.

To use Hive built-in UDFs, see the [LanguageManual UDF](#) on the Apache wiki. To create custom UDFs in Hive, see [Managing Apache Hive User-Defined Functions](#) on page 47.

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. Enter a **Reason for change**, and then 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**.

Using HiveServer2 Web UI in CDH

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 46.

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 HiveServer2 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 HiveServer2 Web UI in Cloudera Manager](#) on page 46.

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 HiveServer2 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 an [Advance 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.

Accessing Apache Hive Table Statistics in CDH

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 Apache Hive User-Defined Functions

You can extend Hive SQL using Java-based user-defined functions (UDFs) and call the UDF from a Hive query.

Prerequisite: In a Hadoop- and Hive-compatible Java project, you write and compile user-defined functionality code into a JAR, and then export the UDF to a JAR. Cloudera recommends that you use the `org.apache.hadoop.hive ql.udf.generic.GenericUDF` API instead when you are creating custom UDFs for Hive. The `org.apache.hadoop.hive ql.exec.UDF` API, which is used for building custom UDFs, is deprecated in CDH 6.0.0 and will be removed in a future release.

In the Registering a UDF procedure below, you store the JAR on your cluster. Using Hive commands, you register the UDF. There are several configuration and registration methods that support ease of use, frequently modified UDFs, and Sentry security. For example, if you use the Hive aux JARs directory method, you need to restart HiveServer2 after registration. If you use another method, you do not need to restart HiveServer2.



Important: This UDF procedure supports the Serializer/Deserializer interface. For example, you can reference SerDes JAR files in table properties by registering the SerDes JAR in the same way as UDF JAR files.

Registering a UDF in Hive

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

You configure the cluster in one of several ways to find the JAR containing your UDF code, and then you register the UDF in Hive.

1. Assuming you just built your Java project in IntelliJ, navigate to the JAR in the /target directory of the project.
2. Choose one of the following methods for configuring the cluster to find the JAR, and then follow the respective step-by-step procedure in sections below:

- [Direct JAR reference configuration](#)

Straight-forward, but recommended for development only. Does not support Sentry.

- [Hive aux JARs directory configuration](#)

Prevents accidental overwriting of files or functions. Recommended for tested, stable UDFs to prevent accidental overwriting of files or functions. Does not support Sentry.

- [Reloadable aux JAR configuration](#)

Avoids HiveServer restarts. Recommended if you anticipate making frequent changes to the UDF logic. Supports Sentry.

If you connect to HiveServer through the load balancer, issuing the RELOAD command loads the JAR file only to the connected HiveServer. Consequently, if you have multiple HiveServer instances behind a load balancer, you must install the JAR file on each node. You also need to connect to each HS2 instance to issue the RELOAD command.

3. After configuring the cluster to find the JAR, use Beeline to start Hive.

- On the command line of a node that runs the HiveServer2 role, type Beeline.
- Use the FQDN of the HiveServer in your cluster to replace myhiveserver.com and enter the database user name and database password, or use the default hive user. For example: `beeline -u jdbc:hive2://myhiveserver.com:10000 -n hive -p`

4. Run one of the following CREATE FUNCTION commands that corresponds to your configuration:

- Direct JAR reference

```
hive> CREATE FUNCTION <function_name> AS '<fully_qualified_class_name>' USING JAR 'hdfs:///<path/to/jar/in/hdfs>';
```

Where the <fully_qualified_class_name> is the full path to the Java class in your JAR file. For example,

```
hive> CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01' USING JAR 'hdfs:///user/max/udf/hiveudf-1.0-SNAPSHOT.jar';
```

- Hive Aux JARs directory

```
hive> CREATE FUNCTION <function_name> AS '<fully_qualified_class_name>'
```

Restart HiveServer2.

- Reloadable Aux JAR

If you use the Reloadable Aux JAR method, RELOAD uploads the JAR to the cluster.

```
hive> RELOAD;
Hive> CREATE FUNCTION <function_name> AS '<fully_qualified_class_name>'
```

For example,

```
hive> RELOAD;
hive> CREATE FUNCTION udftypeof AS 'com.mycompany.hiveudf.TypeOf01';
```

The UDF is registered.

Direct JAR Reference Configuration

You can configure the cluster to find the JAR using the ADD JAR command on the Hive command line.

1. Upload the JAR from your Java project to your CDH cluster. For example, navigate to the JAR in the /target directory of an IntelliJ project, and upload a JAR named TypeOf-1.0-SNAPSHOT.jar.
2. Move the JAR to HDFS.

```
$ sudo su - hdfs
$ hdfs dfs -put TypeOf-1.0-SNAPSHOT.jar /user/max/udf/hiveudf-1.0-SNAPSHOT.jar
```

3. Start Hive from Beeline, and enter the following command:

```
hive> ADD JAR /user/max/udf/hiveudf-1.0-SNAPSHOT.jar
```

The JAR named hiveudf-1.0-SNAPSHOT.jar is added to the Hive classpath.

4. After configuring the cluster, register the UDF as described above.

This configuration method does not work when Beeline runs on a different host from HiveServer2. After using this command, replicate the JAR file to other data nodes in the cluster. You should replicate the JAR to the number of data nodes equal to the square root of the number of cluster nodes.

Hive Aux JARs Directory Configuration

You can configure the cluster to find a UDF JAR by setting `hive.aux.jars.path` to a directory, not a classpath, where you uploaded your JAR. Do not use the `/opt/cloudera/parcels/CDH` directory because that directory changes with every release. You must create and manage the directory on hosts that run Hive metastore, HiveServer2, or the Hive CLI. The directory location is set in the environment as `HIVE_AUX_JARS_PATH` and will generally override the `hive.aux.jars.path` property set in XML files, even if `hive.aux.jars.path` is set in an advanced configuration snippet.

After navigating to the JAR in the /target directory of an IntelliJ project, perform these steps:

1. Upload the JAR file to the host or hosts that run HiveServer2.
2. Give the hive user read, write, and execute access to the directory where the JAR resides, for example `/opt/local/hive/lib/`.
3. If the Hive Metastore runs on a different host or hosts, create the same directory as you created on the HiveServer2 on every Hive Metastore host. For example, create `/opt/local/hive/lib/` on the Hive Metastore host. You do not need to copy the JAR file to the Hive Metastore host directory. Hive takes care of that when you register the UDF. If the directory is not present on the Hive Metastore host, Hive Metastore service does not start.
4. In the **Cloudera Manager Admin Console > Hive service > Configuration** tab.
5. Expand the Hive (Service-Wide) scope.
6. In Advanced, configure the Hive Auxiliary JARs Directory property with the path to the JAR, for example `/opt/local/hive/lib/`.
7. Save changes.
8. In **Cloudera Manager Admin Console > Hive service > Actions**, redeploy the Hive client configuration.
9. Restart the Hive service.
- 10 After configuring the cluster, register the UDF as described above.

Reloadable Aux JAR Configuration

You can configure the cluster to find a UDF JAR by adding the `hive.reloadable.aux.jars.path` property using the Hive Service Advanced Configuration Snippet (Safety Value). For example, after navigating to the JAR in the /target directory of an IntelliJ project, perform these steps:

1. Upload the JAR file to a directory on the a host the runs HiveServer2.

Using & Managing Apache Hive in CDH

2. Give the hive user read, write, and execute access to the directory.
3. If the Hive Metastore runs on a different host or hosts, create the same directory as you created on the HiveServer2 on every Hive Metastore host. For example, create /opt/local/hive/lib/ on the Hive Metastore host. You do not need to copy the JAR file to the Hive Metastore host directory. Hive takes care of that when you register the UDF. If the directory is not present on the Hive Metastore host, Hive Metastore service does not start.
4. In **Cloudera Manager Admin Console > Hive service > Configuration > Filters > Advanced**, click Hive (Service-Wide) scope.
5. In Hive Service Advanced Configuration Snippet (Safety Value) for hive-site.xml, add the following property:
 - name = hive.reloadable.aux.jars.path
 - value = the path to the JAR file
6. Save changes.
7. In **Cloudera Manager Admin Console > Hive service > Actions**, redeploy the Hive client configuration.
8. Restart the Hive service.

This step is only necessary initially. Subsequently, you can add or remove JARs using RELOAD.

9. If you use Sentry, on the Hive command line grant privileges on the JAR files to the roles that require access.

```
GRANT ALL ON URI 'file:///opt/local/hive/lib/<my.jar>' TO ROLE <example_role>;
```

- 10 After configuring the cluster, register the UDF as described above.

Creating Temporary Functions

Use the TEMPORARY keyword in the CREATE FUNCTION command to register a temporary instead of a permanent UDF. For example:

```
hive> CREATE TEMPORARY FUNCTION <your_function_name> AS '<fully_qualified_class_name>'
hive> CREATE TEMPORARY FUNCTION <function_name> AS
'<fully_qualified_class_name>' USING JAR
'hdfs:///<path/to/jar/in/hdfs>';
```

Updating a User-Defined Function

When you change the UDF, you need to re-register it. If you use the ADD JAR method described above, simply drop the function, add the new one, and restart HiveServer2; otherwise, follow these steps to update a UDF:

1. Update your Java code in your Java project.
2. On the Hive command line, drop the UDF that has been updated.

```
hive> DROP FUNCTION hiveudf-1.0-SNAPSHOT.jar;
```

3. Delete the old JAR file from the cluster.
4. Upload the updated JAR to the cluster to the location that corresponds to either your Hive Aux Jars Directory or Reloadable Aux Jars configuration.
5. If Sentry is enabled on your cluster, grant privileges on the JAR files to the roles that require access.
6. If you use Reloadable Aux JARs, start Hive from Beeline, and run the RELOAD command.
7. Run the CREATE FUNCTION command that corresponds to either your Hive Aux Jars Directory or Reloadable Aux Jars configuration as described above.
8. If you use Hive Aux Jars, redeploy the client configurations and restart HiveServer2.

Calling a Hive UDF from Impala

You can call a UDF that you register in Hive from an Impala query under the following conditions:

- Sentry is not enabled.

- When you register the UDF, you use the CREATE FUNCTION that includes the USING CLAUSE (either Direct JAR reference or Hive Aux JARs directory methods).
- Other requirements described in [Impala documentation](#).

The CREATE FUNCTION includes the JAR location; otherwise, Impala does not load the function. Impala relies on the location you provide during function creation. The JAR, which contains the UDF code, must reside on HDFS, making the JAR automatically available to all the Impala nodes. You do not need to manually copy any UDF-related files between servers.

If you cannot register the UDF, which you want to call from Impala, in Hive because, for example, you use Sentry, then register the UDF in Impala. Do not name an Impala-registered UDF the same as any Hive-registered UDF.

Adding Built-in UDFs to the HiveServer2 Blacklist

Built-in UDFs, such as the `year` function, are available in Hive natively. HiveServer2 maintains a blacklist for built-in UDFs to prevent attacks that use the `hive` user credentials to execute Java code. You set the `hive.server2.builtin.udf.blacklist` property to a comma separated list of built-in UDFs that Hive does not execute. A UDF that is included in the blacklist returns an error if invoked from a query. By default this property is empty.

To check whether `hive.server2.builtin.udf.blacklist` contains any UDFs, run the following statement on the Hive command line:

```
hive> SET hive.server2.builtin.udf.blacklist;
```

Any blacklisted UDFs are returned.

To add built-in UDF names to the `hive.server2.builtin.udf.blacklist` property with Cloudera Manager:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. On the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click **HiveServer2** under Scope and click **Advanced** under Category.
4. Search for **HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml** and add the following information:
 - **Name:** `hive.server2.builtin.udf.blacklist`
 - **Value:** `<builtin_udf_name1>,<builtin_udf_name2>...`
 - **Description:** Blacklisted built-in UDFs.
5. Click **Save Changes** and restart the HiveServer2 service for the changes to take effect.

If you are not using Cloudera Manager to manage your cluster, set the `hive.server2.builtin.udf.blacklist` property in the `hive-site.xml` file.

Configuring Transient Apache Hive ETL Jobs to Use the Amazon S3 Filesystem in CDH

Apache 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](#).
- Hive on MapReduce1 is not supported on Amazon S3 in the CDH distribution. Only Hive on MapReduce2/YARN is supported on S3.

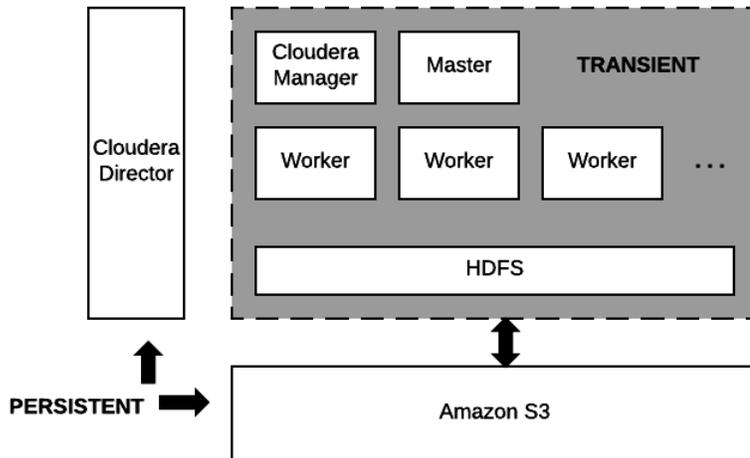
Using & Managing Apache Hive in CDH

For information about how to set up a shared Amazon Relational Database Service (RDS) as your Hive metastore, see [Configuring a Shared Amazon RDS as an HMS for CDH](#) on page 54. For information about tuning Hive read and write performance to the Amazon S3 file system, see [Tuning Apache Hive Performance on the Amazon S3 Filesystem in CDH](#) on page 74.

About Transient Jobs

Most ETL jobs on transient clusters run from scripts that make API calls to a provisioning service such as [Altus 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 Altus 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 Altus 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 [Tuning Hive Write Performance on the Amazon S3 Filesystem](#) for information about tuning Hive to write data to S3 tables or partitions more efficiently.

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 Altus 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 Altus 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 Altus Director

See [Launching an EC2 Instance for Altus Director](#). Install Altus 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 Altus Director.
2. Run the `validate` command by passing the configuration file to it:

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

If Altus Director server is running in a separate instance from the Altus 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.

Run the Altus 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.

Using & Managing Apache Hive in CDH

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-bundle.zip" -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 Altus 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.

Configuring a Shared Amazon RDS as an HMS for CDH

Before CDH 5.10, each CDH cluster had to have its own Apache Hive metastore (HMS) backend database. This model is ideal for clusters where each cluster contains the data locally along with the metadata. In the cloud, however, many

CDH clusters run directly on a shared object store, such as Amazon S3, making it possible for the data to live across multiple clusters and beyond the lifespan of any cluster. In this scenario, clusters need to regenerate and coordinate metadata for the underlying shared data individually.

From CDH 5.10 and later, clusters running in the AWS cloud can share a single persistent instance of the Amazon Relational Database Service (RDS) as the HMS backend database. This enables persistent sharing of metadata beyond a cluster's life cycle so that subsequent clusters need not regenerate metadata as they had to before.

Advantages of This Approach

Using a shared Amazon RDS server as your HMS backend enables you to deploy and share data and metadata across multiple transient as well as persistent clusters if they adhere to restrictions that are outlined in the "Supported Scenarios" section below. For example, you can have multiple transient Hive or Apache Spark clusters writing table data and metadata which can be subsequently queried by a persistent Apache Impala cluster. Or you might have 2-3 different transient clusters, each dealing with different types of jobs on different data sets that spin up, read raw data from S3, do the ETL (Extract, Transform, Load) work, write data out to S3, and then spin down. In this scenario, you want each cluster to be able to simply point to a permanent HMS and do the ETL. Using RDS as a shared HMS backend database greatly reduces your overhead because you no longer need to recreate the HMS again and again for each cluster, every day, for each transient ETL job that you run.

How To Configure Amazon RDS as the Backend Database for a Shared Hive Metastore

The following instructions assumes that you have an Amazon AWS account and that you are familiar with AWS services.

1. Create a MySQL instance with Amazon RDS. See [Creating a MySQL DB Instance...](#) and [Creating an RDS Database Tutorial](#) in Amazon documentation. This step is performed only once. Subsequent clusters that use an existing RDS instance do not need this step because the RDS is already set up.
2. Configure a remote MySQL Hive metastore database as part of the Cloudera Manager installation procedure, using the hostname, username, and password configured during your RDS setup. See [Configuring a Remote MySQL Database for the Hive Metastore](#).
3. Configure Hive, Impala, and Spark to use Amazon S3:
 - For Hive, see [Tuning Hive on S3](#).
 - For Impala, see [Using Impala with the Amazon S3 Filesystem](#).
 - For Spark, see [Accessing Data Stored in Amazon S3 through Spark](#).

Supported Scenarios

The following limitations apply to the jobs you run when you use an RDS server as a remote backend database for Hive metastore.

- No overlapping data or metadata changes to the same data sets across clusters.
- No reads during data or metadata changes to the same data sets across clusters.
- Overlapping data or metadata changes are defined as when multiple clusters concurrently:
 - Make updates to the same table or partitions within the table located on S3.
 - Add or change the same parent schema or database.



Important: If you are running a shared RDS, Cloudera Support will help licensed customers repair any unexpected metadata issues, but will not do "root-cause" analysis.

Configuring ADLS Gen1 Connectivity



Note: This topic discusses Microsoft Azure Data Lake Store (ADLS) Gen 1. For information about ADLS Gen 2, see [Configuring ADLS Gen2 Connectivity](#)

Microsoft Azure Data Lake Store (ADLS) is a massively scalable distributed file system that can be accessed through an HDFS-compatible API. ADLS acts as a persistent storage layer for CDH clusters running on Azure. In contrast to Amazon S3, ADLS more closely resembles native HDFS behavior, providing consistency, file directory structure, and POSIX-compliant ACLs. See the [ADLS documentation](#) for conceptual details.

CDH supports using ADLS as a storage layer for MapReduce2 (MRv2 or YARN), Hive, Hive on Spark, Spark 2.1 and higher, and Spark 1.6. Other applications are not supported and may not work, even if they use MapReduce or Spark as their execution engine. Use the steps in this topic to set up a data store to use with these CDH components.

Note the following limitations:

- ADLS is not supported as the default filesystem. Do not set the default file system property (`fs.defaultFS`) to an `adl://` URI. You can still use ADLS as secondary filesystem while HDFS remains the primary filesystem.
- Hadoop Kerberos authentication is supported, but it is separate from the Azure user used for ADLS authentication.

Setting up ADLS to Use with CDH

1. To create your ADLS account, see the [Microsoft documentation](#).
2. Create the service principal in the Azure portal. See the [Microsoft documentation on creating a service principal](#).



Important:

While you are creating the service principal, write down the following values, which you will need in step 4:

- The client id.
- The client secret.
- The refresh URL. To get this value, in the Azure portal, go to **Azure Active Directory > App registrations > Endpoints**. In the Endpoints region, copy the **OAUTH 2.0 TOKEN ENDPOINT**. This is the value you need for the `refresh_URL` in step 4.

3. Grant the service principal permission to access the ADLS account. See the Microsoft documentation on [Authorization and access control](#). Review the section, "Using ACLs for operations on file systems" for information about granting the service principal permission to access the account.

You can skip the section on RBAC (role-based access control) because RBAC is used for management and you only need data access.

4. Configure your CDH cluster to access your ADLS account. To access ADLS storage from a CDH cluster, you provide values for the following properties when submitting jobs:

Table 1: ADLS Access Properties

Property Description	Property Name
Provider Type	<code>dfs.adls.oauth2.access.token.provider.type</code> The value of this property should be <code>ClientCredential</code>
Client ID	<code>dfs.adls.oauth2.client.id</code>
Client Secret	<code>dfs.adls.oauth2.credential</code>

Property Description	Property Name
Refresh URL	<code>dfs.adls.oauth2.refresh.url</code>

There are several methods you can use to provide these properties to your jobs. There are security and other considerations for each method. Select one of the following methods to access data in ADLS:

- [User-Supplied Key for Each Job](#) on page 57
- [Single Master Key for Cluster-Wide Access](#) on page 58
- [User-Supplied Key stored in a Hadoop Credential Provider](#) on page 58
- [Create a Hadoop Credential Provider and reference it in a customized copy of the core-site.xml file for the service](#) on page 59

Testing and Using ADLS Access

1. After configuring access, test your configuration by running the following command that lists files in your ADLS account:

```
hadoop fs -ls adl://your_account.azuredatalakestore.net/
```

If your configuration is correct, this command lists the files in your account.

2. After successfully testing your configuration, you can access the ADLS account from MRv2, Hive, Hive on Spark, Spark 1.6, Spark 2.1 and higher, or HBase by using the following URI:

```
adl://your_account.azuredatalakestore.net
```

For additional information and examples of using ADLS access with Hadoop components:

- **Spark:** See [Accessing Data Stored in Azure Data Lake Store \(ADLS\) through Spark](#)
- **distcp:** See [Using DistCp with Microsoft Azure \(ADLS\)](#).
- **TeraGen:**

```
export HADOOP_CONF_DIR=path_to_working_directory
export HADOOP_CREDSTORE_PASSWORD=hadoop_credstore_password
hadoop jar /opt/cloudera/parcels/CDH/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
teragen 1000 adl://jzhugeadls.azuredatalakestore.net/tg
```

User-Supplied Key for Each Job

You can pass the ADLS properties on the command line when submitting jobs.

- **Advantages:** No additional configuration is required.
- **Disadvantages:** Credentials will appear in log files, command history and other artifacts, which can be a serious security issue in some deployments.



Important: Cloudera recommends that you only use this method for access to ADLS in development environments or other environments where security is not a concern.

Use the following syntax to run your jobs:

```
hadoop command
-Ddfs.adls.oauth2.access.token.provider.type=ClientCredential \
-Ddfs.adls.oauth2.client.id=CLIENT_ID \
-Ddfs.adls.oauth2.credential='CLIENT_SECRET' \
-Ddfs.adls.oauth2.refresh.url=REFRESH_URL \
```

```
adl://<store>.azuredatalakestore.net/src hdfs://nn/tgt
```

Single Master Key for Cluster-Wide Access

Use Cloudera Manager to save the values in the **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml**.

- **Advantages:** All users can access the ADLS storage
- **Disadvantages:** This is a highly insecure means of providing access to ADLS for the following reasons:
 - The credentials will appear in all Cloudera Manager-managed configuration files for all services in the cluster.
 - The credentials will appear in the Job History server.



Important: Cloudera recommends that you only use this method for access to ADLS in development environments or other environments where security is not a concern.

1. Open the Cloudera Manager Admin Console and go to **Cluster Name > Configuration > Advanced Configuration Snippets**.
2. Enter the following in the **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml**:

```
<property>
  <name>dfs.adls.oauth2.access.token.provider.type</name>
  <value>ClientCredential</value>
</property>
<property>
  <name>dfs.adls.oauth2.client.id</name>
  <value>CLIENT_ID</value>
</property>
<property>
  <name>dfs.adls.oauth2.credential</name>
  <value>CLIENT_SECRET</value>
</property>
<property>
  <name>dfs.adls.oauth2.refresh.url</name>
  <value>REFRESH_URL</value>
</property>
```

3. Click **Save Changes**.
4. Click **Restart Stale Services** so the cluster can read the new configuration information.

User-Supplied Key stored in a Hadoop Credential Provider

- **Advantages:** Credentials are securely stored in the credential provider.
- **Disadvantages:** Works with MapReduce2 and Spark only (Hive, Impala, and HBase are not supported).

1. Create a [Credential Provider](#).
 - a. Create a password for the Hadoop Credential Provider and export it to the environment:

```
export HADOOP_CREDSTORE_PASSWORD=password
```

- b. Provision the credentials by running the following commands:

```
hadoop credential create dfs.adls.oauth2.client.id -provider
jceks://hdfs/user/USER_NAME/adls-cred.jceks -value client ID
hadoop credential create dfs.adls.oauth2.credential -provider
jceks://hdfs/user/USER_NAME/adls-cred.jceks -value client secret
hadoop credential create dfs.adls.oauth2.refresh.url -provider
jceks://hdfs/user/USER_NAME/adls-cred.jceks -value refresh URL
```

You can omit the `-value` option and its value and the command will prompt the user to enter the value.

For more details on the `hadoop credential` command, see [Credential Management \(Apache Software Foundation\)](#).

2. Reference the Credential Provider on the command line when submitting jobs:

```
hadoop command
  -Ddfs.adls.oauth2.access.token.provider.type=ClientCredentialial \
-Dhadoop.security.credential.provider.path=jceks://hdfs/user/USER_NAME/adls-cred.jceks
\
  adl://<store>.azuredatalakestore.net/
```

Create a Hadoop Credential Provider and reference it in a customized copy of the `core-site.xml` file for the service

- **Advantages:** all users can access the ADLS storage
- **Disadvantages:** you must pass the path to the credential store on the command line.

1. Create a [Credential Provider](#):

- a. Create a password for the Hadoop Credential Provider and export it to the environment:

```
export HADOOP_CREDSTORE_PASSWORD=password
```

- b. Provision the credentials by running the following commands:

```
hadoop credential create dfs.adls.oauth2.client.id -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value client ID
hadoop credential create dfs.adls.oauth2.credential -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value client secret
hadoop credential create dfs.adls.oauth2.refresh.url -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value refresh URL
```

You can omit the `-value` option and its value and the command will prompt the user to enter the value.

For more details on the `hadoop credential` command, see [Credential Management \(Apache Software Foundation\)](#).

2. Copy the contents of the `/etc/service/conf` directory to a working directory. The *service* can be one of the following verify list:

- yarn
- spark
- spark2

Use the `--dereference` option when copying the file so that symlinks are correctly resolved. For example:

```
cp -r --dereference /etc/spark/conf ~/my_custom_config_directory
```

Change the ownership so that you can edit the files:

```
sudo chown --recursive $USER ~/custom-conf-file/*
```

3. Add the following to the `core-site.xml` file in the working directory:

```
<property>
  <name>hadoop.security.credential.provider.path</name>
  <value>jceks://hdfs/path_to_credential_store_file</value>
```

```
</property>
<property>
  <name>dfs.adls.oauth2.access.token.provider.type</name>
  <value>ClientCredential</value>
</property>
```

The value of the `path_to_credential_store_file` should be the same as the value for the `--provider` option in the `hadoop credential create` command described in step 1.

4. Set the `HADOOP_CONF_DIR` environment variable to the location of the working directory:

```
export HADOOP_CONF_DIR=path_to_working_directory
```

Creating a Credential Provider for ADLS

You can use a Hadoop Credential Provider to specify ADLS credentials, which allows you to run jobs without having to enter the access key and secret key on the command line. This prevents these credentials from being exposed in console output, log files, configuration files, and other artifacts. Running the command in this way requires that you provision a credential store to securely store the access key and secret key. The credential store file is saved in HDFS.

To create a credential provider, run the following commands:

1. Create a password for the Hadoop Credential Provider and export it to the environment:

```
export HADOOP_CREDSTORE_PASSWORD=password
```

2. Provision the credentials by running the following commands:

```
hadoop credential create dfs.adls.oauth2.client.id -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value client ID
hadoop credential create dfs.adls.oauth2.credential -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value client secret
hadoop credential create dfs.adls.oauth2.refresh.url -provider
jceks://hdfs/user/USER_NAME/adlskeyfile.jceks -value refresh URL
```

You can omit the `-value` option and its value and the command will prompt the user to enter the value.

For more details on the `hadoop credential` command, see [Credential Management \(Apache Software Foundation\)](#).

ADLS Configuration Notes

ADLS Trash Folder Behavior

If the `fs.trash.interval` property is set to a value other than zero on your cluster and you do not specify the `-skipTrash` flag with your `rm` command when you remove files, the deleted files are moved to the trash folder in your ADLS account. The trash folder in your ADLS account is located at `adl://your_account.azuredatalakestore.net/user/user_name/.Trash/current/`. For more information about HDFS trash, see [Configuring HDFS Trash](#).

User and Group Names Displayed as GUIDs

By default ADLS user and group names are displayed as GUIDs. For example, you receive the following output for these Hadoop commands:

```
$hadoop fs -put /etc/hosts adl://your_account.azuredatalakestore.net/one_file
$hadoop fs -ls adl://your_account.azuredatalakestore.net/one_file
-rw-r--r--  1 94c1b91f-56e8-4527-b107-b52b6352320e cdd5b9e6-b49e-4956-be4b-7bd3ca314b18
  273
2017-04-11 16:38 adl://your_account.azuredatalakestore.net/one_file
```

To display user-friendly names, set the property `adl.feature.ownerandgroup.enableupn` to `true` in the `core-site.xml` file or at the command line. When this property is set to `true` the `-ls` command returns the following output:

```
$hadoop fs -ls adl://your_account.azuredatalakestore.net/one_file
-rw-r--r--  1 YourADLSApp your_login_app    273 2017-04-11 16:38
adl://your_account.azuredatalakestore.net/one_file
```

Importing Data into Hive with Sqoop Through HiverServer2

Importing Data Through Hiveserver2

In addition to importing with the Hive CLI, Sqoop supports import into Hive through HiveServer2 as well.

There are three HiveServer2 specific command options that the user can define for the `sqoop import` tool when importing data:

- `--hs2-url`: The JDBC connection string to HiveServer2 as one would specify it for Beeline.
- `--hs2-user`: Specifies the user for creating the JDBC connection to HiveServer2. If a user is not specified, the current OS user is used by default.
- `--hs2-keytab`: The path to the keytab file of the user connecting to HiveServer2. If the `--hs2-user` option is specified then `--hs2-keytab` option has to be specified as well otherwise it can be omitted. The keytab has to be available on the machine the Sqoop command is executed on.

HiveServer2 imports can be initiated with the `--hs2-url`. When the user specifies the `--hs2-url` option, commands are sent to HiveServer2 through a JDBC connection. The data itself is not transferred via the JDBC connection. It is written directly to HDFS and moved to the Hive warehouse using the `LOAD DATA INPATH` command just like in the case of the default Hive import. When the `--hs2-url` option is not specified, Sqoop imports the data into Hive using the Hive CLI, which is the default method. For more information regarding the default import method, see upstream documents.

HiveServer2 provides proper Sentry authorization. As a result, Cloudera recommends importing data into Hive through HiveServer2 instead of the default method. Currently, Sqoop can authenticate to HiveServer2 using Kerberos only.

Importing Data

Prerequisites

Before importing data make sure that the following prerequisites are satisfied:

- A properly configured user with permissions to execute `CREATE TABLE` and `LOAD DATA INPATH` statements in Hive.
- Default ACLs defined for the temporary import folder so that the new folder, when created, inherits the ACLs of the parent.

Steps

1. Create a temporary import folder with read, write, and execute permissions for the Hive user. For example:

```
hdfs dfs -mkdir /user/username/importdir
hdfs dfs -setfacl -m default:user:hive:rwx /user/username/importdir
```

The `LOAD DATA INPATH` statement is executed by the Hive superuser, therefore, the temporary HDFS folder that Sqoop imports into has to have read, write, and execute permission for the Hive user as well.



Important: Make sure that effective ACLs are not constrained for the Hive user by the `fs.permissions.umask-mode` setting.

2. Execute a Hive import. Use either of the following methods:

a. Execute a Hive import with the current OS user:

```
sqoop import --connect $MYCONN --username $MYUSER --password $MYPSWD --table
"employees_test" --target-dir "/user/username/importdir/employees_test" --hive-import
--hs2-url
"jdbc:hs2://host:1000/default,principalhive/host@DOMAIN,ssl=true,sslTrustStore=/etc/ssl/cf/CACERT/truststore.js,trustStorePassword'
```

b. Execute a Hive import with the username and keytab specified:

```
sqoop import --connect $MYCONN --username $MYUSER --password $MYPSWD --table
"employees_test" --target-dir "/user/username/importdir/employees_test"
--delete-target-dir --hive-import --hs2-url
"jdbc:hs2://host:1000/default,principalhive/host@DOMAIN,ssl=true,sslTrustStore=/etc/ssl/cf/CACERT/truststore.js,trustStorePassword'
--hs2-user username --hs2-keytab "/path/to/sqooptestkeytab"
```

Tuning Apache Hive in CDH

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.

For more information about tuning Hive, see [Tuning Apache Hive Performance on the Amazon S3 Filesystem in CDH](#) on page 74.

Heap Size and Garbage Collection for Hive Components

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

Memory and Hardware Requirements 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. The table below contains guidelines for sizing the heap for each component, based on your cluster size. The table refers to connections, the number of open connections to HiveServer (Cloudera Manager `hive_open_connections` metric). In Cloudera Manager, HiveServer2, Status, the visual representation of this metric appears. For example:



Component	Java Heap		CPU	Disk
HiveServer 2	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required for the following: <ul style="list-style-type: none"> • HiveServer2 log files • <code>stdout</code> and <code>stderr</code> output files • Configuration files
	2-10 connections	4-6 GB		
	11-20 connections	6-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		

Component	Java Heap		CPU	Disk
	<p>Cloudera recommends splitting HiveServer2 into multiple instances and load balancing them once you start allocating more than 16 GB to HiveServer2. The objective is to adjust the size to reduce the impact of Java garbage collection on active processing by the service.</p> <p>Set this value using the Java Heap Size of HiveServer2 in Bytes Hive configuration property.</p> <p>For more information, see Tuning Hive in CDH.</p>			<ul style="list-style-type: none"> • Operation logs stored in the <code>operation_logs_dir</code> directory, which is configurable • Any temporary files that might be created by local map tasks under the <code>/tmp</code> directory
Hive Metastore	Single Connection	4 GB	Minimum 4 dedicated cores	Minimum 1 disk This disk is required so that the Hive metastore can store the following artifacts: <ul style="list-style-type: none"> • Logs • Configuration files • Backend database that is used to store metadata if the database server is also hosted on the same node
	2-10 connections	4-10 GB		
	11-20 connections	10-12 GB		
	21-40 connections	12-16 GB		
	41 to 80 connections	16-24 GB		
	<p>Set this value using the Java Heap Size of Hive Metastore Server in Bytes Hive configuration property.</p> <p>For more information, see Tuning Hive in CDH.</p>			
Beeline CLI	Minimum: 2 GB	N/A	N/A	

 **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, set option `MaxMetaspaceSize` to put an upper limit on the amount of native memory used for class metadata.

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

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

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:

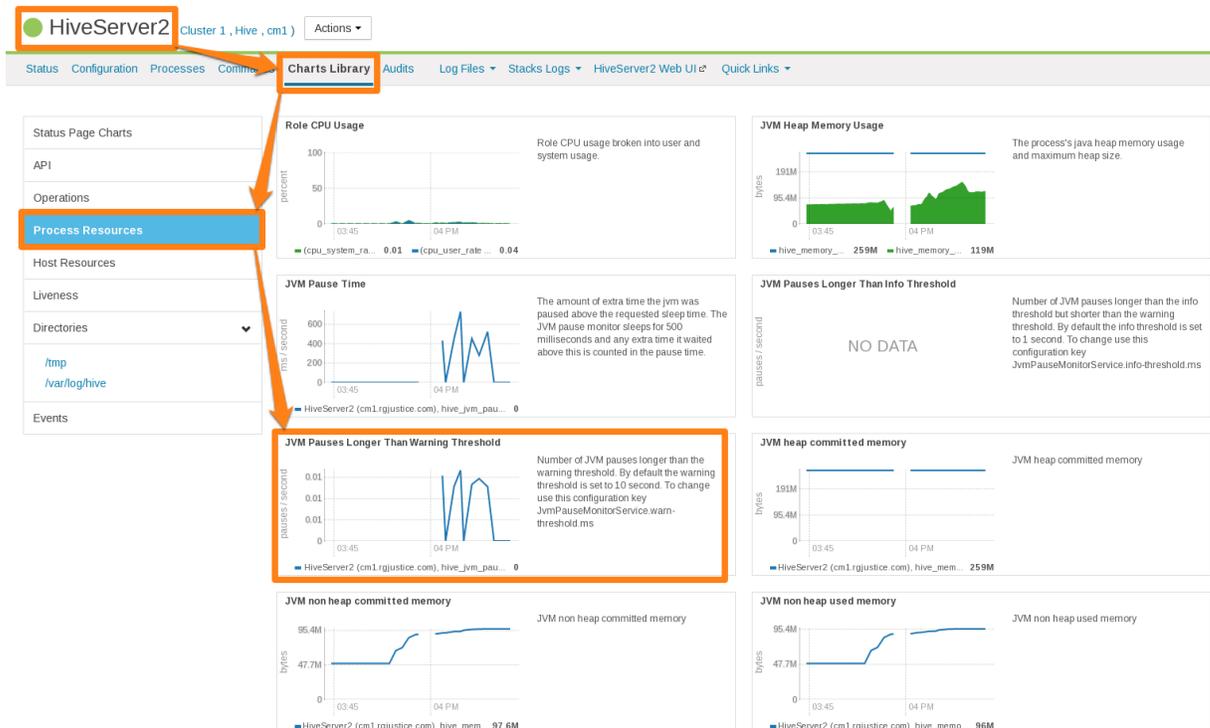


Figure 2: Cloudera Manager Chart Library Page for Process Resources

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 68.

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 later, 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 68 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 68.
- 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 82 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](#).
- In the Metastore Server Advanced Configuration Snippet (Safety Valve) for hive-site.xml, 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.
 - Increase the cluster-wide limit beyond 1000, if needed. This action 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)**

Tuning Apache Hive in CDH

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.

• Queries fail with "Too many counters" error

Hive operations use various counters while executing MapReduce jobs. These per-operator counters are enabled by the configuration setting `hive.task.progress`. This is disabled by default. If it is enabled, Hive might create a large number of counters (4 counters per operator, plus another 20).



Note: If dynamic partitioning is enabled, Hive implicitly enables the counters during data load.

By default, CDH restricts the number of MapReduce counters to 120. Hive queries that require more counters fail with the "Too many counters" error.

How to resolve:

– For managed clusters:

1. In Cloudera Manager Admin Console, go to the MapReduce service.
2. Select the **Configuration** tab.
3. Type **counters** in the search box in the right panel.
4. Scroll down the right panel to locate the `mapreduce.job.counters.max` property and increase the **Value**.
5. Click **Save Changes**.

– For unmanaged clusters:

Set the `mapreduce.job.counters.max` property to a higher value in `mapred-site.xml`.

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.

Tuning Apache Hive on Spark in CDH



Note: This page contains references to CDH 5 components or features that have been removed from CDH 6. These references are only applicable if you are managing a CDH 5 cluster with Cloudera Manager 6. For more information, see [Deprecated Items](#).

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`. The default for `spark.yarn.executor.memoryOverhead` is `executorMemory * 0.10`, with minimum of 384. This property represents the off-heap memory to be allocated per executor (units = MB). The off-heap memory is used for VM overheads, interned strings, and other overhead, and increases proportionately with the executor size between 5-10%.

If you need to set a value different from the default value, the following example shows how to dynamically set properties in the Hive session:

```
set spark.executor.memory=12g;
set spark.yarn.executor.memoryOverhead=2g;
```

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.yarn.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 Apache 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
```

Tuning Apache Hive in CDH

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 number 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.

Tuning Apache Hive Performance on the Amazon S3 Filesystem in CDH

Some of the default behaviors of Apache Hive might degrade performance when reading and writing data to tables stored on Amazon S3. Cloudera has introduced the following enhancements that make using Hive with S3 more efficient.

Tuning Hive Write Performance on S3

Hive can write the final job in the query plan in parallel to the S3 file system. HiveServer2 uses a thread pool of workers to transfer the data to the final table location on S3. The default values of tuning parameters generally yield good performance for a wide range of workloads. However, if necessary, you can further tune the parameters to optimize for specific workloads.

Hive S3 Write Performance Tuning Parameters

To improve write performance for Hive tables stored on S3, use Cloudera Manager to set the parameters listed below. See [Setting Parameters as Service-Wide Defaults with Cloudera Manager](#) on page 75.

Parameter Name	Description	Settings	Default
<code>hive.mv.files.thread</code>	 Important: Only tune this parameter when you have confirmed that thread pool parallelism is impacting performance. Before making any changes, contact Cloudera Support for guidance.	Range between: 0 and 40	15

Parameter Name	Description	Settings	Default
	<p>Sets the number of threads used to move files in a move task. Increasing the value of this parameter increases the number of parallel copies that can run on S3.</p> <p>A separate thread pool is used for each Hive query. When running only a few queries in parallel, you can increase this parameter for greater per-query write throughput. However, when you run a large number of queries in parallel, decrease this parameter to avoid thread exhaustion.</p> <p>To disable multi-threaded file moves, set this parameter to 0. This can prevent thread contention on HiveServer2.</p> <p>This parameter also controls renames on HDFS, so increasing this value increases the number of threads responsible for renaming files on HDFS.</p>		
hive.blobstore.use.blobstore.as.scratchdir	<p>When set to <code>true</code>, this parameter enables the use of scratch directories directly on S3.</p> <div style="border: 1px solid orange; padding: 5px; margin-top: 10px;"> <p> Important: Enabling this parameter might degrade performance slightly, but is useful if the HDFS cluster is not large enough to hold the intermediate data from a Hive query.</p> </div>	true false	false

Setting Parameters on a Per-Query Basis with the Hive SET Command

Optimize on a per-query basis by setting these parameters in the query code with the Hive `SET` command.

For example, to set the thread pool to 20 threads and enable scratch directories on S3:

```
set hive.mv.files.thread=20
set hive.blobstore.use.blobstore.as.scratchdir=true
```

Setting Parameters as Service-Wide Defaults with Cloudera Manager

Use Cloudera Manager to set `hive.mv.files.thread` and `hive.blobstore.use.blobstore.as.scratchdir` as service-wide defaults:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope.
4. Click the **Performance** category.
5. Search for each parameter to set them.
6. Click **Save Changes**.

Tuning the S3A Connector to Improve Hive Write Performance on S3

The `fs.s3a` parameters are used to tune the S3A Connector inside the Hadoop code base. The S3A Connector configurations control the number of threads used to issue concurrent upload and copy requests. A single instance of

the S3A Connector is used with a HiveServer2 instance, so different Hive queries can share the same connector instance. The same thread pool is used to issue upload and copy requests. This means that the `fs.s3a` parameters cannot be set on a per-query basis. Instead, set them for each HiveServer2 instance. In contrast, the thread pool controlled by `hive.mv.files.thread` is created for each query separately.

Parameter Name	How To Tune
<code>fs.s3a.threads.core</code>	Increase the value to increase the number of core threads in the thread pool used to run any data uploads or copies.
<code>fs.s3a.threads.max</code>	Increase the value to increase the maximum number of concurrent active partition uploads and copies, which each use a thread from the thread pool.
<code>fs.s3a.max.total.tasks</code>	Increase the value to increase the number of partition uploads and copies allowed to the queue before rejecting additional uploads.
<code>fs.s3a.connection.maximum</code>	Increase the value to increase the maximum number of simultaneous connections to S3. Cloudera recommends setting this value to 1500.

Setting S3A Connector Parameters as Service-Wide Defaults

Use Cloudera Manager to set the S3A Connector parameters as service-wide defaults for Hive:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope.
4. Click the **Advanced** category.
5. Search for the **HiveServer2 Advanced Configuration Snippet (Safety Valve) for hive-site.xml** configuration setting and click the plus sign to add parameters.
6. For each `fs.s3a` parameter, type the parameter name into the **Name** field and the value in the **Value** field.
7. Click **Save Changes**.

Known Limitations

1. If you have a large number of concurrent Hive query operations running, a deadlock might occur in the `S3AFileSystem` class of the Hadoop platform. This is caused by thread pool limits and causes HiveServer2 to freeze. If this occurs, you must restart HiveServer2. To work around the issue, increase the values of `fs.s3a.threads.core` and `fs.s3a.threads.max`. See [HADOOP-13826](#).

This behavior might occur more frequently if `fs.s3a.blocking.executor.enabled` is set to `true`. This parameter is turned off by default in CDH.
2. S3 is an eventually consistent storage system. See the [S3 documentation](#). This eventual consistency affects Hive behavior on S3 and, in rare cases, can cause intermittent failures. Retrying the failed query usually works around the issue.

Tuning Hive Dynamic Partitioning Performance on S3

[Dynamic partitioning](#) is a Hive feature that enables dynamic insertions of data into partitions based on the value of a column in a record. It is useful for bulk creating or updating partitions. Prior to CDH 5.11, performance of Hive queries that performed dynamic partitioning on S3 was diminished because partitions were loaded into the target table one at a time. In CDH 5.11 and later, optimizations change the underlying logic so that partitions are loaded in parallel.

Use the following parameter to tune performance on a wide range of workloads that use dynamic partitioning. This parameter can be set with Cloudera Manager at the service level or on a per-query basis using the Hive `SET` command. See [Setting the Hive Dynamic Partition Loading Parameter as a Service-Wide Default with Cloudera Manager](#) on page 77.

Parameter Name	Description	Settings	Default
<code>hive.load.dynamic.partitions.thread</code>	<p>Sets the number of threads used to load dynamically generated partitions.</p> <p>Loading dynamically generated partitions requires renaming the files to their destination location and updating the new partition metadata.</p> <p>Increasing the value set for this parameter can improve performance when you have several hundred dynamically generated partitions.</p>	Range between: 0 and 25	15

Tuning Tips

Increase the value set for `hive.load.dynamic.partitions.thread` to improve dynamic partitioning query performance on S3. However, do not set this parameter to values exceeding 25 to avoid placing an excessive load on S3, which can lead to throttling issues.

Setting the Hive Dynamic Partition Loading Parameter on a Per-Query Basis

Optimize dynamic partitioning at the session level by using the Hive `SET` command in the query code.

For example, to set the thread pool to 25 threads:

```
set hive.load.dynamic.partitions.thread=25
```

Setting the Hive Dynamic Partition Loading Parameter as a Service-Wide Default with Cloudera Manager

Use Cloudera Manager to set `hive.load.dynamic.partitions.thread` as a service-wide default:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope.
4. Click the **Performance** category.
5. Search for **Load Dynamic Partitions Thread Count** and enter the value you want to set as a service-wide default.
6. Click **Save Changes**.

Tuning Hive INSERT OVERWRITE Performance on S3

`INSERT OVERWRITE` queries write data to a specific table or partition, overwriting any existing data. When Hive detects existing data in the target directory, it moves the existing data to the [HDFS trash directory](#). Moving data to the trash directory can significantly degrade performance when it is run on S3. In CDH 5.11 and later, an optimization is added to move data to the trash directory in parallel by using the following parameter. Use Cloudera Manager to set this parameter as a service-wide default or use the Hive `SET` command to set the parameter on a per-query basis. See [Setting the Hive INSERT OVERWRITE Performance Tuning Parameter as a Service-Wide Default with Cloudera Manager](#) on page 78.



Important: This optimization only applies to `INSERT OVERWRITE` queries that insert data into tables or partitions where already there is existing data.

Parameter Name	Description	Settings	Default
<code>hive.mv.files.thread</code>	Set this parameter to control the number of threads used to delete existing data in the HDFS trash directory for <code>INSERT OVERWRITE</code> queries.	Range between: 0 and 40	15

Parameter Name	Description	Settings	Default
	 Important: Originally, this parameter only controlled the number of threads used by HiveServer2 to move data from the staging directory to another location. This parameter can also be used to tune Hive write performance on S3 tables. See Hive S3 Write Performance Tuning Parameters on page 74.		

Tuning Tips

The `hive.mv.files.thread` parameter can be tuned for `INSERT OVERWRITE` performance in the same way it is tuned for write performance. See [Hive S3 Write Performance Tuning Parameters](#) on page 74.

If setting the above parameter does not produce acceptable results, you can disable the HDFS trash feature by setting the `fs.trash.interval` to 0 on the HDFS service. In Cloudera Manager, choose **HDFS > Configuration > NameNode > Main** and set **Filesystem Trash Interval** to 0.

 **Warning:** Disabling the trash feature of HDFS causes permanent data deletions, making the deleted data unrecoverable.

Setting the Hive INSERT OVERWRITE Performance Tuning Parameter on a Per-Query Basis

Configure Hive to move data to the HDFS trash directory in parallel for `INSERT OVERWRITE` queries using the Hive `SET` command.

For example, to set the thread pool to use 30 threads at a maximum:

```
set hive.mv.files.thread=30
```

Setting the Hive INSERT OVERWRITE Performance Tuning Parameter as a Service-Wide Default with Cloudera Manager

Use Cloudera Manager to set `hive.mv.files.thread` as a service-wide default:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope.
4. Click the **Performance** category.
5. Search for **Move Files Thread Count** and enter the value you want to set as a service-wide default.
6. Click **Save Changes**.

Tuning Hive Table Partition Read Performance on S3

Prior to CDH 5.11, Hive queries that read over 1,000 partitions stored on S3 experienced performance degradation because metadata operations against S3 are much slower than metadata operations performed against HDFS. When Hive runs a query, it needs to collect metadata about the files and about the directory it is reading from. This metadata includes information such as number of files or file sizes. To collect this metadata, Hive must make calls to S3. Before CDH 5.11, these metadata calls were issued serially (one at a time). In CDH 5.11 and later, the metadata operations have been optimized so that the calls are now issued in parallel. This optimization delivers the most benefit for queries that read from multiple partitions. Benefits for queries that read from non-partitioned tables are less significant.

Use the following parameters to tune Hive table partition read performance on S3. The default values yield good performance gains for a wide range of workloads, but you can further tune them to optimize for specific workloads.

These parameters can be set with Cloudera Manager at the service level or on a per-query basis using the Hive `SET` command. See [Setting Hive Table Partition Read Performance Tuning Parameters as Service-Wide Defaults with Cloudera Manager](#) on page 79.

Parameter Name	Description	Settings	Default
<code>hive.exec.input.listing.max.threads</code>	Sets the maximum number of threads that Hive uses to list input files. Increasing this value can improve performance when there are many partitions being read.	Range between: 0 and 50	15
<code>mapreduce.input.fileinputformat.list-status.num-threads</code>	Sets the number of threads used by the <code>FileInputFormat</code> class when listing and fetching block locations for the specified input paths.	Range between: 0 and 50	1

Tuning Tips

If listing input files becomes a bottleneck for the Hive query, increase the values for `hive.exec.input.listing.max.threads` and `mapreduce.input.fileinputformat.list-status.num-threads`. This bottleneck might occur if the query takes a long time to list input directories or to run split calculations when reading several thousand partitions. However, do not set these parameters to values over 50 to avoid putting excessive load on S3, which might lead to throttling issues.

Setting the Hive Table Partition Read Performance Tuning Parameters on a Per-Query Basis

Configure Hive to perform metadata collection in parallel when reading table partitions on S3 using the Hive `SET` command.

For example, to set the maximum number of threads that Hive uses to list input files to 20 and the number of threads used by the `FileInputFormat` class when listing and fetching block locations for input to 5:

```
set hive.exec.input.listing.max.threads=20
set mapreduce.input.fileinputformat.list-status.num-threads=5
```

Setting Hive Table Partition Read Performance Tuning Parameters as Service-Wide Defaults with Cloudera Manager

Use Cloudera Manager to set `hive.exec.input.listing.max.threads` and `mapreduce.input.fileinputformat.list-status.num-threads` as service-wide defaults.

To set `hive.exec.input.listing.max.threads`:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, click the **HiveServer2** scope.
4. Click the **Performance** category.
5. Search for **Input Listing Max Threads** and enter the value you want to set as a service-wide default.
6. Click **Save Changes**.

To set `mapreduce.input.fileinputformat.list-status.num-threads`:

1. In the Cloudera Manager Admin Console, go to the MapReduce service.
2. In the MapReduce service page, click the **Configuration** tab.
3. Search for **MapReduce Service Advanced Configuration Snippet (Safety Valve) for `mapred-site.xml`** and enter the parameter, value, and description:

```
<property>
  <name>mapreduce.input.fileinputformat.list-status.num-threads</name>
  <value>number_of_threads</value>
```

```
<description>Number of threads used to list and fetch block locations for input paths
specified by FileInputFormat</description>
</property>
```

4. Click **Save Changes**.

Tuning Hive MSCK (Metastore Check) Performance on S3

Running the `MSCK` command with the `REPAIR TABLE` option is a simple way to bulk add partitions to Hive tables. See the [Apache Language Manual](#) for details about using `MSCK REPAIR TABLE`. `MSCK REPAIR TABLE` scans the file system to look for directories that correspond to a partition and then registers them with the Hive metastore. Prior to CDH 5.11, `MSCK` performance was slower on S3 when compared to HDFS due to the overhead created by collecting metadata on S3. In CDH 5.11 and later, `MSCK` metadata calls are now issued in parallel, which significantly improves performance.

Use the following parameters to tune Hive `MSCK` metadata call performance on S3. The default values yield good performance gains for a wide range of workloads, but you can further tune them to optimize for specific workloads. The `hive.metastore.fshandler.threads` parameter can be set as a service-wide default with Cloudera Manager, but cannot be set at the session level. The `hive.msck.repair.batch.size` parameter can be set with Cloudera Manager at the service level or on a per-query basis using the Hive `SET` command. See [Setting the Hive MSCK REPAIR TABLE Tuning Parameters as Service-Wide Defaults with Cloudera Manager](#) on page 81.

Parameter Name	Description	Settings	Default
<code>hive.metastore.fshandler.threads</code>	Sets the number of threads that the Hive metastore uses when adding partitions in bulk to the metastore. Each thread performs metadata operations for each partition added, such as collecting statistics for the partition or checking if the partition directory exists. This parameter is also used to control the size of the thread pool that is used by <code>MSCK</code> when it scans the file system looking for directories that correspond to table partitions. Each thread performs a list status on each possible partition directory.	Range between: 0 and 30	15
<code>hive.msck.repair.batch.size</code>	Sets the number of partition objects sent per batch from the HiveServe2 service to the Hive metastore service with the <code>MSCK REPAIR TABLE</code> command. If this parameter is set to a value higher than zero, new partition information is sent from HiveServer2 to the Hive metastore in batches. Sending this information in batches improves how memory is used in the metastore, avoiding client read timeout exceptions. If this parameter is set to 0, all partition information is sent at once in a single Thrift call.	Range between: 0 and 2,147,483,647	0

Tuning Tips

The `hive.metastore.fshandler.threads` parameter can be increased if the `MSCK REPAIR TABLE` command is taking excessive time to scan S3 for potential partitions to add. Do not set this parameter to a value higher than 30 to avoid putting excessive load on S3, which can lead to throttling issues.

Increase the value set for the `hive.msck.repair.batch.size` parameter if you receive the following exception:

```
SocketTimeoutException: Read timed out
```

This exception is thrown by HiveServer2 when a metastore operation takes longer to complete than the time specified for the `hive.metastore.client.socket.timeout` parameter. If you simply increase the timeout, it must be set across all metastore operations and requires restarting the metastore service. It is preferable to increase the value set for `hive.msck.repair.batch.size`, which specifies the number of partition objects that are added to the metastore at one time. Increasing `hive.msck.repair.batch.size` to 3000 can help mitigate timeout exceptions returned when running `MSCK` commands. Set to a lower value if you have multiple `MSCK` commands running in parallel.

Setting `hive.msck.repair.batch.size` on a Per-Query Basis

Use the Hive `SET` command to specify how many partition objects are sent per batch from the HiveServer2 service to the Hive metastore service at the session level.

For example, to specify that batches containing 3,000 partition objects each are sent:

```
set hive.msck.repair.batch.size=3000
```

Setting the Hive `MSCK REPAIR TABLE` Tuning Parameters as Service-Wide Defaults with Cloudera Manager

Use Cloudera Manager to set the `hive.metastore.fshandler.threads` and the `hive.msck.repair.batch.size` parameters as service-wide defaults:

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. In the Hive service page, click the **Configuration** tab.
3. On the Configuration page, search for each parameter to set them.
4. Click **Save Changes**.

Configuring HMS High Availability in CDH

You can enable Hive metastore high availability (HA) so that your cluster is resilient to failures if a metastore becomes unavailable. The HA mode is recommended to address fail-over situations. No load balancing is done.

Recommendations

Cloudera recommends that each instance of the metastore runs on a separate cluster host.

Enabling HMS 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.

To apply this configuration property to other role groups as needed, 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.

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:

- In the first step of enabling HiveServer2 high availability below, you enable Hive Delegation Token Store implementation. Oozie needs this implementation for secure HS2 HA. Otherwise, the Oozie server can get a delegation token from one HS2 server, but the actual query might run against another HS2 server, which does not recognize the HS2 delegation token. Exception: If you enable HMS HA, do not enable Hive Delegation Token Store; otherwise, Oozie job issues occur.
- 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's principal to connect to HS2 directly; otherwise, 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. 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.

To apply this configuration property to other role groups as needed, 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`.
3. Add multiple HiveServer2 instances to your cluster:
 1. Click the **Instances** tab, and then click **Add Role Instances**.

2. 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.
 3. Click **OK**, and then click **Continue**. The Instances page appears where you can start the new HiveServer2 instances.
 4. Click the **Configuration** tab.
 5. Select **Scope > HiveServer2**.
 6. Select **Category > Main**.
 7. Locate the **HiveServer2 Load Balancer** property or search for it by typing its name in the Search box.
 8. 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.
9. Click **Save Changes** to commit the changes.
 - 10 **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-hmac-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

If you are not already using a load-balancing proxy, you can experiment with HAProxy a free, open source load balancer.

To install and configure HAProxy, an open source load balancer, perform the following steps.

1. Download the appropriate from the HAProxy web site.
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
```

Query Vectorization for Apache Hive in CDH

By default, the Hive query execution engine processes one row of a table at a time. The single row of data goes through all the operators in the query before the next row is processed, resulting in very inefficient CPU usage. In vectorized query execution, data rows are batched together and represented as a set of column vectors. The basic idea of vectorized query execution is to process a batch of rows as an array of column vectors:

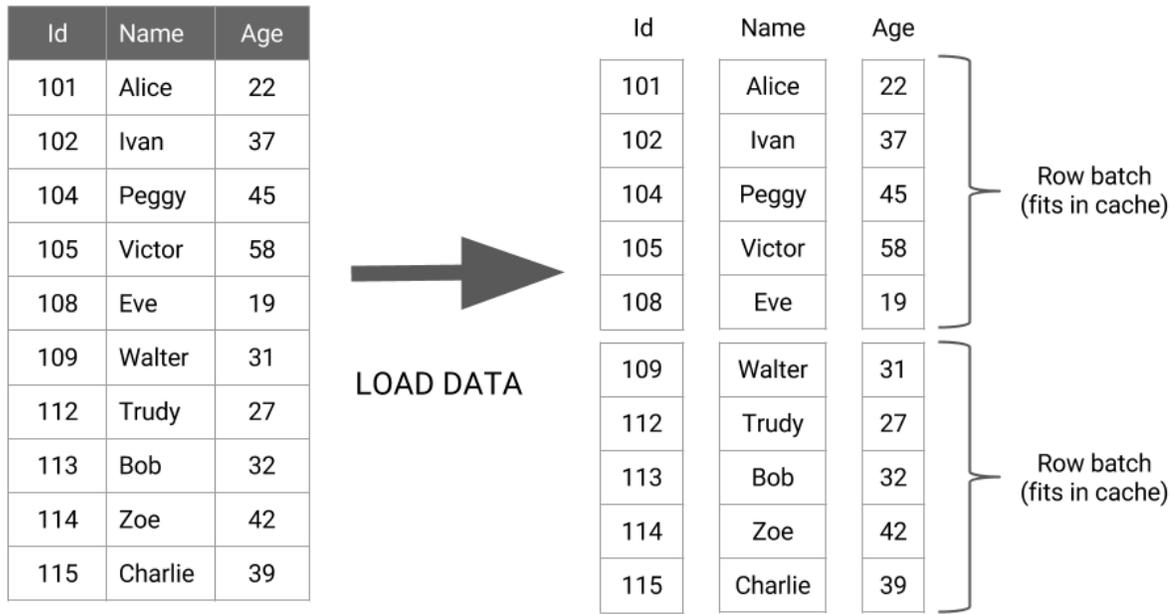


Figure 3: How Hive Query Vectorization Works

When query vectorization is enabled, the query engine processes vectors of columns, which greatly improves CPU utilization for typical query operations like scans, filters, aggregates, and joins.

Enabling Hive Query Vectorization

Hive query vectorization is enabled by default in CDH 6 and CDH 5. However, in CDH 5 vectorized query execution in Hive is only possible on ORC-formatted tables, which Cloudera recommends you do not use for overall compatibility with the CDH platform. Instead, Cloudera recommends that you use tables in the Parquet format because all CDH components support this format and it can be consumed by all CDH components.

Hive query vectorization is enabled or disabled for *all* file formats by setting the `hive.vectorized.execution.enabled` property to `true` or `false` *and* making sure that no value is set for the `hive.vectorized.input.format.excludes` property. To ensure that query vectorization is used for the Parquet file format, you must make sure that the `hive.vectorized.input.format.excludes` property is not set to `org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat`.

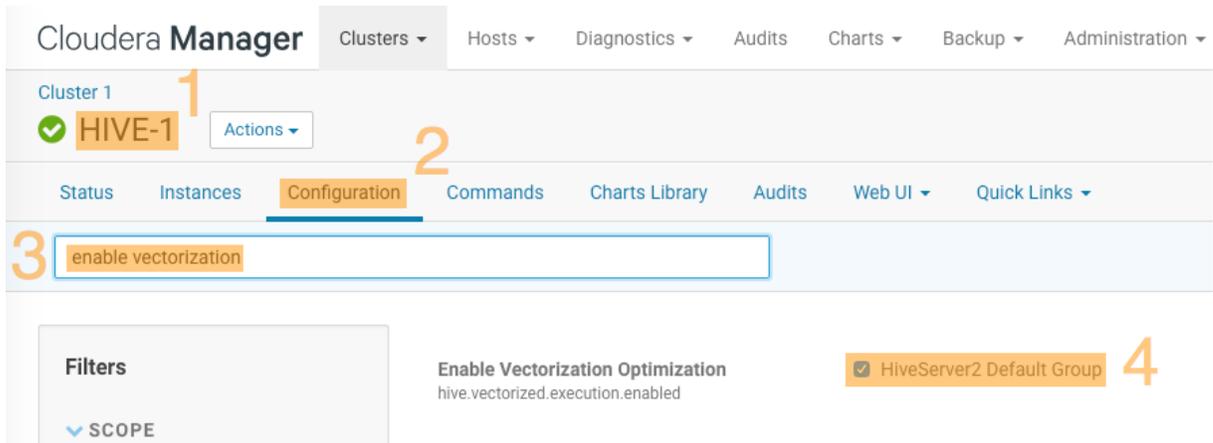
Using Cloudera Manager to Enable or Disable Query Vectorization for Parquet Files on a Server-wide Basis

For managed clusters, open the Cloudera Manager Admin Console and perform the following steps:

1. Select the Hive service.
2. Click the **Configuration** tab.
3. Search for `enable vectorization`.

To view all the available vectorization properties for Hive, search for `hiveserver2_vectorized`. All the vectorization properties are in the **Performance** category.

4. Select the **Enable Vectorization Optimization** option to enable query vectorization. To disable query vectorization, uncheck the box that is adjacent to **HiveServer2 Default Group**.

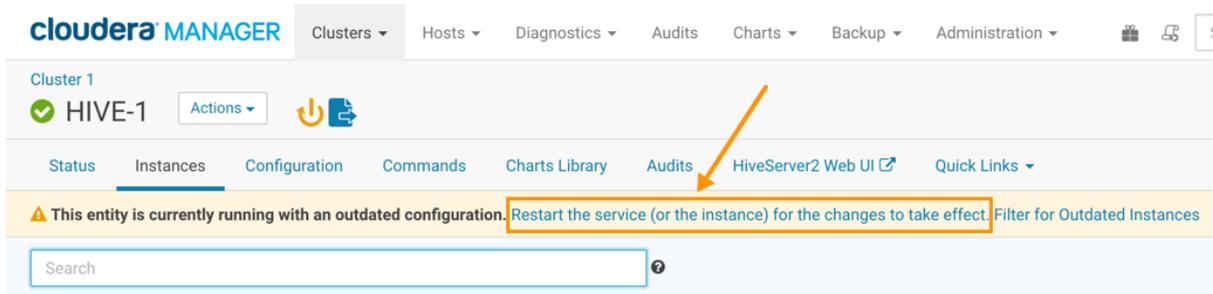


5. To enable or disable Hive query vectorization for the Parquet file format, set the **Exclude Vectorized Input Formats** property in Cloudera Manager as follows:

- To *disable* vectorization for Parquet files only, set this property to **org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat**
- To *enable* vectorization for all file formats including Parquet, set this property to **Custom** and leave the setting blank.

6. Click **Save Changes**.

7. Click the **Instances** tab, and then click the **Restart the service (or the instance) for the changes to take effect**:



Manually Enabling or Disabling Query Vectorization for Parquet Files on a Server-Wide Basis

To **enable** query vectorization for Parquet files on unmanaged clusters on a server-wide basis:

- Set the `hive.vectorized.execution.enabled` property to `true` in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.execution.enabled</name>
  <value>>true</value>
  <description>Enables query vectorization.</description>
</property>
```

- Ensure there is no value set for the `hive.vectorized.input.format.excludes` property in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.input.format.excludes</name>
  <value/>
  <description>Does not exclude query vectorization on any file format including Parquet.</description>
```

```
</property>
```

To *disable* query vectorization for *Parquet files only* on unmanaged clusters on a server-wide basis:

- Set the `hive.vectorized.execution.enabled` property to `true` in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.execution.enabled</name>
  <value>true</value>
  <description>Enables query vectorization.</description>
</property>
```

- Set the `hive.vectorized.input.format.excludes` property to `org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat` in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.input.format.excludes</name>
  <value>org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat</value>
  <description>Disables query vectorization on Parquet file formats only.</description>
</property>
```

To *enable* query vectorization on *all* file formats:

- Set the `hive.vectorized.execution.enabled` property to `true` in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.execution.enabled</name>
  <value>true</value>
  <description>Enables query vectorization.</description>
</property>
```

- Ensure there is no value set for the `hive.vectorized.input.format.excludes` property in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.input.format.excludes</name>
  <value/>
  <description>Does not exclude query vectorization on any file format.</description>
</property>
```

To *disable* query vectorization on *all* file formats:

Set the `hive.vectorized.execution.enabled` property to `false` in the `hive-site.xml` file:

```
<property>
  <name>hive.vectorized.execution.enabled</name>
  <value>false</value>
  <description>Disables query vectorization on all file formats.</description>
</property>
```

Enabling or Disabling Hive Query Vectorization for Parquet Files on a Session Basis

Use the Hive `SET` command to enable or disable query vectorization on an individual session. Enabling or disabling query vectorization on a session basis is useful to test the effects of vectorization on the execution of specific sets of queries.

To enable query vectorization for all file formats including Parquet on an individual session only:

```
SET hive.vectorized.execution.enabled=true;
SET hive.vectorized.input.format.excludes= ;
```

Setting `hive.vectorized.input.format.excludes` to a blank value ensures that this property is unset and that no file formats are excluded from query vectorization.

To disable query vectorization for Parquet files only on an individual session only:

```
SET hive.vectorized.execution.enabled=true;
SET
hive.vectorized.input.format.excludes=org.apache.hadoop.hive ql.io.parquet.MapredParquetInputFormat ;
```

To enable query vectorization for all file formats on an individual session only:

```
SET hive.vectorized.execution.enabled=true;
SET hive.vectorized.input.format.excludes= ;
```

Setting `hive.vectorized.input.format.excludes` to a blank value ensures that this property is unset and that no file formats are excluded from query vectorization.

To disable query vectorization for all file formats on an individual session only:

```
SET hive.vectorized.execution.enabled=false;
```

Tuning Hive Query Vectorization

When query vectorization is enabled, there are additional properties you can set to tune how your queries are vectorized. These properties can be set in Cloudera Manager, can be set manually in the `hive-site.xml` file, or can be set on a per-query basis using the Hive `SET` command. Use the same general steps listed in [the previous section](#) to configure these properties in Cloudera Manager or manually.

hive.vectorized.adaptor.usage.mode

Description: Specifies the extent to which the vectorization engine tries to vectorize UDFs that do not have native vectorized versions available. Selecting the `none` option specifies that only queries using native vectorized UDFs are vectorized. Selecting the `chosen` option specifies that Hive chooses to vectorize a subset of the UDFs based on performance benefits using the Vectorized Adaptor. Selecting the `all` option specifies that the Vectorized Adaptor be used for all UDFs even when native vectorized versions are not available.

Recommendations: For optimum stability and correctness of query output, set this option to `chosen`.

Default Setting: `chosen`

hive.vectorized.execution.reduce.enabled

Description: Turns on or off vectorization for the reduce-side of query execution. Applies only when the execution engine is set to Spark.

Recommendations: Enable this property by setting it to `true` if you are using Hive on Spark. Otherwise, do not enable this property.

Default Setting: `true`

hive.vectorized.groupby.checkinterval

Description: For vectorized `GROUP BY` operations, specifies the number of row entries added to the hash table before rechecking the average variable size when estimating memory usage.

Recommendations: Current testing indicates that the default setting is applicable in most cases.

Default Setting: `4096`

hive.vectorized.groupby.flush.percent

Description: Sets the percentage between 0 and 100 percent of entries in the vectorized `GROUP BY` aggregation hash that is flushed when the memory threshold is exceeded. To set no flushing, set this property to `0.0`. To set flushing at 100 percent, set this property to `1.0`.

Recommendations: This sets the amount of data that is held in memory. To increase performance, increase the setting. However, increase the setting conservatively to prevent out-of-memory issues.

Default Setting: `0.1`, which sets the flush percentage to 10%

hive.vectorized.input.format.excludes

Description: Specifies input formats to exclude from vectorized query execution. You can select `org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat`, `org.apache.hadoop.hive.ql.io.orc.OrcInputFormat`, or **Custom** from a drop-down list. When you select **Custom**, you can add another input format for exclusion, but currently no other format is supported. If you select **Custom** and leave the field blank, query vectorization is applied to all file formats.



Important: Vectorized execution can still occur for an excluded input format based on whether row SerDes or vector SerDes are enabled.

Recommendations: Use this property to automatically disable certain file formats from vectorized execution. Cloudera recommends that you test your workloads on development clusters using vectorization and enable it in production if you receive significant performance advantages. As an example, if you want to exclude vectorization only on the ORC file format while keeping vectorization for all other file formats including the Parquet file format, set this property to `org.apache.hadoop.hive.ql.io.orc.OrcInputFormat`.

Default Setting: `org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat` which disables query vectorization for the Parquet file format only.

hive.vectorized.use.checked.expressions

Description: To enhance performance, vectorized expressions operate using wide data types like long and double. When wide data types are used, numeric overflows can occur during expression evaluation in a different manner for vectorized expressions than they do for non-vectorized expressions. Consequently, different query results can be returned for vectorized expressions compared to results returned for non-vectorized expressions. When enabled, Hive uses vectorized expressions that handle numeric overflows in the same way as non-vectorized expressions are handled.

Recommendations: Keep this property set to `true`, if you want results across vectorized and non-vectorized queries to be consistent.

Default Setting: `true`

hive.vectorized.use.vectorized.input.format

Description: Enables Hive to take advantage of input formats that support vectorization when they are available.

Recommendations: Enable this property by setting it to `true` if you have Parquet or ORC workloads that you want to be vectorized.

Default Setting: `true`

hive.vectorized.use.vector.serde.deserialize

Description: Enables Hive to use built-in vector SerDes to process text and SequenceFile tables for vectorized query execution. In addition, this configuration also helps vectorization of intermediate tasks in multi-stage query execution.

Recommendations: Keep this set to `false`. Setting this property to `true` might help multi-stage workloads, but when set to `true`, it enables text vectorization, which Cloudera does not support.

Default Setting: `false`

Supported/Unsupported Data Types and Functions

Most common data types and functions are supported by Hive query vectorization on Parquet tables in CDH. The following subsections provide more details about data type and function support.

Supported/Unsupported Data Types

Currently, some complex data types, such as `map`, `list`, and `union` are not supported for Hive query vectorization on Parquet tables in CDH. Even though the `struct` data type is supported, it is vectorized only when all of the fields defined within the `struct` are primitives. The following data types are supported.

Table 2: Supported Data Types for Hive Query Vectorization on Parquet Tables

<code>int</code>	<code>smallint</code>	<code>tinyint</code>
<code>bigint</code>	<code>integer</code>	<code>long</code>
<code>short</code>	<code>timestamp</code>	<code>interval_year_month</code>
<code>boolean</code>	<code>binary</code>	<code>string</code>
<code>byte</code>	<code>float</code>	<code>double</code>
<code>void</code>	<code>struct</code>	

Supported/Unsupported Functions

Common arithmetic, boolean (for example `AND`, `OR`), comparison, mathematical (for example `SIN`, `COS`, `LOG`), date, and type-cast functions are supported. Also common aggregate functions such as `MIN`, `MAX`, `COUNT`, `AVG`, and `SUM` are also supported. If a function is not supported, the vectorizer attempts to vectorize the function based on the configuration value specified for `hive.vectorized.adaptor.usage.mode`. You can set this property to `none` or `chosen`. To set this property in Cloudera Manager, search for the `hive.vectorized.adaptor.usage.mode` property on the Configuration page for the Hive service, and set it to `none` or `chosen` as appropriate. For unmanaged clusters, set it manually in the `hive-site.xml` file for server-wide scope. To set it on a session basis, use the Hive `SET` command as described [above](#).

Verifying a Query is Vectorized

To verify that a query is vectorized, use the `EXPLAIN VECTORIZATION` statement. This statement returns a query plan that shows how the Hive query execution engine processes your query and whether vectorization is being triggered.

Example of Verifying that Query Vectorization is Triggered for Your Query

This example uses the Hive table `p_clients`, which uses the Parquet format and contains the following columns and data types:

```
DESCRIBE p_clients;
```

col_name	data_type	comment
name	string	
symbol	string	
lastsale	double	
marketlabel	string	
marketamount	bigint	
ipoyear	int	
segment	string	
business	string	
quote	string	

To get the query execution plan for a query, enter the following commands in a Beeline session:

```
EXPLAIN VECTORIZATION SELECT COUNT(*) FROM p_clients WHERE ipoyear = 2009;
```

This command returns the following query execution plan:

```

-----
                Explain
-----
PLAN VECTORIZATION: 1
  enabled: true
  enabledConditionsMet: [hive.vectorized.execution.enabled IS true] |

STAGE DEPENDENCIES:
  Stage-1 is a root stage
  Stage-0 depends on stages: Stage-1

STAGE PLANS: 2
  Stage: Stage-1
    Map Reduce
      Map Operator Tree:
        TableScan
          alias: p_clients
          filterExpr: (ipoyear = 2009) (type: boolean) |
          Statistics: Num rows: 2804 Data size: 25236 Basic stats: COMPLETE Column stats: NONE |
        Filter Operator
          predicate: (ipoyear = 2009) (type: boolean) |
          Statistics: Num rows: 1402 Data size: 12618 Basic stats: COMPLETE Column stats: NONE |
        Select Operator
          Statistics: Num rows: 1402 Data size: 12618 Basic stats: COMPLETE Column stats: NONE |
        Group By Operator
          aggregations: count()
          mode: hash
          outputColumnNames: _col0
          Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: NONE |
        Reduce Output Operator
          sort order:
            Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: NONE |
            value expressions: _col0 (type: bigint) |
      Execution mode: vectorized
      Map Vectorization: 3
        enabled: true
        enabledConditionsMet: hive.vectorized.use.vectorized.input.format IS true |
        groupByVectorOutput: true
        inputFileFormats: org.apache.hadoop.hive.ql.io.parquet.MapredParquetInputFormat |
        allNative: false
        usesVectorUDFAdaptor: false
        vectorized: true
      Reduce Vectorization: 4
        enabled: false
        enableConditionsNotMet: hive.vectorized.execution.reduce.enabled IS false, hive.execution.engine mr IN [tez, spark] IS false |
      Reduce Operator Tree:
        Group By Operator
          aggregations: count(VALUE._col0)
          mode: mergepartial
          outputColumnNames: _col0
          Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: NONE |
        File Output Operator
          compressed: false
          Statistics: Num rows: 1 Data size: 8 Basic stats: COMPLETE Column stats: NONE |
          table:
            input format: org.apache.hadoop.mapred.SequenceFileInputFormat |
            output format: org.apache.hadoop.hive.ql.io.HiveSequenceFileOutputFormat |
            serde: org.apache.hadoop.hive.serde2.lazy.LazySimpleSerDe |

  Stage: Stage-0
    Fetch Operator
      limit: -1
      Processor Tree:
        ListSink
-----

```

Figure 4: EXPLAIN VECTORIZATION Query Execution Plan for Hive Table Using the Parquet Format

Vectorization is explained in several parts of this query plan:

1. The `PLAN VECTORIZATION` section shows a high-level view of the vectorization status for the query. The `enabled` flag set to `true` means that vectorization is turned on and the `enabledConditionsMet` flag shows that it is

enabled because the `hive.vectorized.execution.enabled` property is set to `true`. If vectorization is not enabled, the `enabledConditionsNotMet` flag shows why.

2. Then in the `STAGE PLANS` section, the output shows the vectorization status for each task of query execution. For example, there might be multiple map and reduce tasks for a query and it is possible that only a subset of these tasks are vectorized. In the above example, the `Stage-1` sub-section shows there is only one map task and one reduce task. The `Execution mode` sub-section of the map task shows whether the task is vectorized. In this case, `vectorized` displays, which means that the vectorizer was able to successfully validate and vectorize all of the operators for this map task.
3. The `Map Vectorization` sub-section shows more details of map task vectorization. Specifically, the configurations that affect the map side vectorization are shown along with whether these configurations are enabled. If the configurations are enabled, they are listed for `enabledConditionsMet`. If the configurations are not enabled, they are listed for `enabledConditionsNotMet` as explained in the above `PLAN VECTORIZATION` section. In this example, it shows that the map side of query execution is enabled because the `hive.vectorized.use.vectorized.input.format` property is set to `true`. This section also contains details about input file format and adaptor settings used in the map side of query execution.
4. The `Reduce Vectorization` sub-section shows that the reduce side of query execution was not vectorized because the `hive.vectorized.execution.reduce.enabled` property is set to `false`. This sub-section also shows that the execution engine is not set to Tez or Spark, which are needed for reduce side vectorization. In this particular example, to enable reduce side vectorization, the execution engine should be set to Spark and the `hive.vectorized.execution.reduce.enabled` property should be set to `true`.

By using the `EXPLAIN VECTORIZATION` statement with your queries, you can find out before you deploy them whether vectorization will be triggered and what properties you must set to enable it.

Hive/Impala Replication



Note: This page contains references to CDH 5 components or features that have been removed from CDH 6. These references are only applicable if you are managing a CDH 5 cluster with Cloudera Manager 6. For more information, see [Deprecated Items](#).

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

Hive/Impala 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/Impala 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/Impala 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 with Kerberos Authentication](#) for details about configuring it.



Note: If your deployment includes tables backed by Kudu, BDR filters out Kudu tables for a Hive replication in order to prevent data loss or corruption.

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/Impala Replication

If your cluster has Hive non-Gateway roles installed on hosts with limited resources, Hive/Impala 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/Impala 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/Impala replication. For example:

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

5. Enter a **Reason for change**, and then 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.

Replication of Parameters

Parameters of databases, tables, partitions, and indexes are replicated by default during Hive/Impala replications.

You can disable replication of parameters:

1. Log in to the Cloudera Manager Admin Console.
2. Go to the Hive service.
3. Click the **Configuration** tab.
4. Search for "Hive Replication Environment Advanced Configuration Snippet"
5. Add the following parameter:

```
REPLICATE_PARAMETERS=false
```

6. Click **Save Changes**.

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.

Guidelines for Snapshot Diff-based Replication

By default, BDR uses snapshot differences ("diff") to improve performance by comparing HDFS snapshots and only replicating the files that are changed in the source directory. While Hive metadata requires a full replication, the data stored in Hive tables can take advantage of snapshot diff-based replication.

To use this feature, follow these guidelines:

- The source and target clusters must be managed by Cloudera Manager 5.15.0 or higher. If the destination is Amazon S3 or Microsoft ADLS, the source cluster must be Managed by Cloudera Manager 5.15.0 or higher. Snapshot diff-based restore to S3 or ADLS is not supported
- The source and target clusters run CDH 5.15.0 or higher, 5.14.2 or higher, or 5.13.3 or higher.
- Verify that HDFS snapshots are immutable.

In the Cloudera Manager Admin Console, go to **Clusters > <HDFS service> > Configuration** and search for **Enable Immutable Snapshots**.

- Do not use snapshot diff for globbed paths. It is not optimized for globbed paths.
- Set the snapshot root directory as low in the hierarchy as possible.
- Decide if you want BDR to abort on a snapshot diff failure or continue the replication. If you choose to configure BDR to continue the replication when it encounters an error, BDR performs a complete replication. Note that continuing the replication can result in a longer duration since a complete replication is performed.
- BDR performs a complete replication when one or more of the following change: Delete Policy, Preserve Policy, Target Path, or Exclusion Path.
- Paths from *both* source and destination clusters in the replication schedule must be under a snapshottable root or should be snapshottable for the schedule to run using snapshot diff.
- If a Hive replication schedule is created to replicate a database, ensure all the HDFS paths for the tables in that database are either snapshottable or under a snapshottable root. For example, if the database that is being replicated has external tables, all the external table HDFS data locations should be snapshottable too. Failing to do so will cause BDR to fail to generate a diff report. Without a diff report, BDR will not use snapshot diff.
- After every replication, BDR retains a snapshot on the source cluster. Using the snapshot copy on the source cluster, BDR performs incremental backups for the next replication cycle. BDR retains snapshots on the source cluster only if:
 - Source and target clusters in the Cloudera Manager are 5.15 and higher
 - Source and target CDH are 5.13.3+, 5.14.2+, and 5.15+ respectively

Replicating from Insecure to Secure Clusters

You can use BDR to replicate data from an insecure cluster, one that does not use Kerberos authentication, to a secure cluster, a cluster that uses Kerberos. Note that the reverse is not true. BDR does not support replicating from a secure cluster to an insecure cluster.

To perform the replication, the destination cluster must be managed by Cloudera Manager 6.1.0 or higher. The source cluster must run Cloudera Manager 5.14.0 or higher in order to be able to replicate to Cloudera Manager 6. For more information about supported replication scenarios, see [Supported Replication Scenarios](#).



Note: In replication scenarios where a destination cluster has multiple source clusters, all the source clusters must either be secure or insecure. BDR does not support replication from a mixture of secure and insecure source clusters.

To enable replication from an insecure cluster to a secure cluster, you need a user that exists on all the hosts on both the source cluster and destination cluster. Specify this user in the **Run As Username** field when you create a replication schedule.

The following steps describe how to add a user:

1. On a host in the source or destination cluster, add a user with the following command:

```
sudo -u hdfs hdfs dfs -mkdir -p /user/<username>
```

For example, the following command creates a user named `milton`:

```
sudo -u hdfs hdfs dfs -mkdir -p /user/milton
```

2. Set the permissions for the user directory with the following command:

```
sudo -u hdfs hdfs dfs -chown <username> /user/<username>
```

For example, the following command makes `milton` the owner of the `milton` directory:

```
sudo -u hdfs hdfs dfs -chown milton /user/milton
```

3. Create the `supergroup` group for the user you created in step 1 with the following command:

```
groupadd supergroup
```

4. Add the user you created in step 1 to the group you created:

```
usermod -G supergroup <username>
```

For example, add `milton` to the group named `supergroup`:

```
usermod -G supergroup milton
```

5. Repeat this process for all hosts in the source and destination clusters so that the user and group exists on all of them.

After you complete this process, specify the user you created in the **Run As Username** field when you create a replication schedule.

Configuring Replication of Hive/Impala 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](#).
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 replicating to or from S3 or ADLS, follow the steps under [Hive/Impala Replication To and From Cloud Storage](#) on page 106 before completing these steps.

- a. Use the **Name** field to provide a unique name for the replication schedule.
- b. Use the **Source** drop-down list to select the cluster with the Hive service you want to replicate.

- c. 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.
- d. Based on the type of destination cluster you plan to use, select one of these two options:
- Use HDFS Destination
 - Use Cloud Destination



Note: For using cloud storage in the target cluster, you must set up a valid cloud storage account and verify that the cloud storage has enough space to save the replicated data.

- e. If you select **Use Cloud Destination**, provide the following details:
- **Cloud Destination Account** - Enter the name of the cloud account which is used to store the replicated data. For example: ADLS or S3. This should be one of the configured external accounts.
 - **Cloud Destination Path** - Specifies the cloud destination storage path where you can store the replicated data from source to destination cluster. The Hive2 instance is updated to point its metadata to use the cloud destination where the replicated data resides.
- f. 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
<code>[\w] . +</code>	Any database or table name.
<code>(?!myname\b) . +</code>	Any database or table except the one named myname.
<code>db1 db2 [\w_]+</code>	All tables of the db1 and db2 databases.
<code>db1 [\w_]+</code> Click the "+" button and then enter <code>db2 [\w_]+</code>	All tables of the db1 and db2 databases (alternate method).

- g. 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.
- h. 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.

- i. Specify the **Run on peer as Username** option if the peer cluster is configured with a different superuser. This is only applicable while working in a kerberized environment.
6. Select the **Resources** tab to configure the following:
 - **Scheduler Pool** – (Optional) Enter the name of a resource pool in the 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`
 - **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 up to and including 5.11.x, select the **Replicate Impala Metadata** drop-down list and select **No** to avoid redundant replication of Impala metadata. (This option only displays when supported by both source and destination clusters.) You can select the following options for **Replicate Impala Metadata**:
 - **Yes** – replicates the Impala metadata.
 - **No** – does not replicate the Impala metadata.
 - **Auto** – Cloudera Manager determines whether or not to replicate the Impala metadata based on the CDH version.

To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.7 or lower, see [Replicating Data to Impala Clusters](#) 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/Impala replication process detects incompatible changes on the source cluster, Hive/Impala 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.

- **Number of concurrent HMS connections** - The number of concurrent Hive Metastore connections. These connections are used to concurrently import and export metadata from Hive. Increasing the number of threads can improve BDR performance. By default, any new replication schedules will use 5 connections.

If you set the value to 1 or more, BDR uses multi-threading with the number of connections specified. If you set the value to 0 or fewer, BDR uses single threading and a single connection.

Note that the source and destination clusters must run a Cloudera Manager version that supports concurrent HMS connections, Cloudera Manager 5.15.0 or higher and Cloudera Manager 6.1.0 or higher.

- 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.
- **Description** - A description for the replication schedule.
- **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.
- **Skip Listing Checksum Checks** - Whether to skip checksum check when comparing two files to determine whether they are same or not. If skipped, the file size and last modified time are used to determine if files are the same or not. Skipping the check improves performance during the mapper phase. Note that if you select the **Skip Checksum Checks** option, this check is also skipped.
- **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.
- **Abort on Snapshot Diff Failures** - If a snapshot diff fails during replication, BDR uses a complete copy to replicate data. If you select this option, the BDR aborts the replication when it encounters an error instead.
- **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 S3 or ADLS.)
 - **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 102.

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/Impala replications. See [User-Defined Functions \(UDFs\)](#), [Replicating Data to Impala Clusters](#), and [Managing Apache Hive User-Defined Functions](#) on page 47.

To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.6 or lower, see [Replicating Data to Impala Clusters](#) for information on when to select the **Replicate Impala Metadata** option on the **Advanced** tab when creating a Hive/Impala 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

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.

For previously-run replications, the number of replicated UDFs displays on the **Replication History** page:

Replication History (Replication Schedules)

Type	Start Time	Duration	Outcome	Tables	Files Expected	Files Copied	Files Failed	FI
HIVE	June 30, 2016 4:42 PM	1 min	Successful	1	2 (4.6 MiB)	0 (0 B)	0 (0 B)	FI

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.

Search							
Actions for Selected ▾						Create Schedule ▾	Last Refreshed 9:08 AM
<input type="checkbox"/>	ID	Type	Source	Destination	Last Run	Next Run	Actions ▾
<input type="checkbox"/>	4	HDFS	HDFS-1 Cluster 1 @ n57u	HDFS-1 Cluster 1	✓ 9:06 AM	None scheduled.	Actions ▾
Message: 0 file(s) copied, 0 unchanged. From: /user/hue To: /user/hue_b							
<input type="checkbox"/>	5	Hive	HIVE-1 Cluster 1 @ n57u	HIVE-2 Cluster 2	● None	📅 06/07/2016	Actions ▾
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 3: 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.
Name	The unique name you specify when you create a schedule.
Type	The type of replication scheduled, either HDFS or Hive.
Source	The source cluster for the replication.
Destination	The destination cluster for the replication.
Throughput	Average throughput per mapper/file of all the files written. Note that throughput does not include the following information: the combined throughput of all mappers and the time taken to perform a checksum on a file after the file is written.
Progress	The progress of the replication.
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: <ul style="list-style-type: none"> ✓ - 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.  60%  - Running. Displays a spinner and bar showing the progress of the replication.

Column	Description										
	Click the Last Run column label to sort the Replication Schedules table by the last run date.										
Next Run	<p>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.</p> <p>Click the Next Run column label to sort the Replication Schedules table by the next run date.</p>										
Objects	<p>Displays on the bottom line of each row, depending on the type of replication:</p> <ul style="list-style-type: none"> • Hive - A list of tables selected for replication. • HDFS - A list of paths selected for replication. <p>For example:</p> <table border="1"> <thead> <tr> <th><input type="checkbox"/></th> <th>ID</th> <th>Type</th> <th>Source</th> <th>Destination</th> </tr> </thead> <tbody> <tr> <td><input type="checkbox"/></td> <td>4</td> <td>HDFS</td> <td>HDFS-1 Cluster 1 @ n57u</td> <td>HDFS-1 Cluster 1</td> </tr> </tbody> </table> <p>Message: HDFS replication command succeeded. From: /user/hue To: /user/hue_b</p>	<input type="checkbox"/>	ID	Type	Source	Destination	<input type="checkbox"/>	4	HDFS	HDFS-1 Cluster 1 @ n57u	HDFS-1 Cluster 1
<input type="checkbox"/>	ID	Type	Source	Destination							
<input type="checkbox"/>	4	HDFS	HDFS-1 Cluster 1 @ n57u	HDFS-1 Cluster 1							
Actions	<p>The following items are available from the Action button:</p> <ul style="list-style-type: none"> • 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: <ol style="list-style-type: none"> 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:

- Click **Actions > Enable | Disable | Delete** in the row for a replication schedule.

To enable, disable, or delete multiple replication schedules:

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	Start Time	Duration	Outcome	Files Expected	Files Copied	Files Failed	Files Deleted	Files Skipped
Type HDFS Source HDFS-1 (Cluster 1) @ n56u Destination HDFS-1 (Cluster 1) Next Run None scheduled.	May 23, 2016 10:04 AM Started At May 23, 2016 10:04 AM Duration a few seconds Command Details View Diagnostics Collect Diagnostic Data	1 min	Successful	0 (0 B)	0 (0 B)	0 (0 B)	0	0 (0 B)
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	Start Time	Duration	Outcome	Tables	Files Expected	Files Copied	Files Failed	Files Deleted	Files Skipped
Type HIVE Source HIVE-1 (Cluster 1) Destination HIVE-2 (Cluster 2) Next Run None scheduled	September 25, 2015 11:54 AM 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	0 min	Failed	1	-	-	-	-	-
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 4: Replication History Table

Column	Description
Start Time	Time when the replication job started. Expand the display and show details of the replication. In this screen, you can: <ul style="list-style-type: none"> • Click the View link to open the Command Details page, which displays details and messages about each step in the execution of the command. Expand the display for a Step to:

Column	Description
	<ul style="list-style-type: none"> – 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. <p>See Viewing Running and Recent Commands.</p> <ul style="list-style-type: none"> • 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: <ol style="list-style-type: none"> 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/Impala 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.
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/Impala Replication To and From Cloud Storage

You can use Cloudera Manager to replicate Hive/Impala data and metadata to and from S3 or ADLS, however you cannot replicate data from one S3 or ADLS instance to another using Cloudera Manager. You must have the appropriate

credentials to access the S3 or ADLS account. Additionally, you must create buckets in S3 or a data lake store in ADLS to store the replicated files.

When you replicate data to cloud storage with BDR, BDR also backs up file metadata, including extended attributes and ACLs.

To configure Hive/Impala replication to or from S3 or ADLS:

1. Create **AWS Credentials** or **Azure Credentials**. See [How to Configure AWS Credentials](#) or [Configuring ADLS Access Using Cloudera Manager](#).



Important: If AWS S3 access keys are rotated, you must restart Cloudera Manager server; otherwise, Hive replication fails.

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 S3 or ADLS destination (one of the **AWS Credentials** or **ADLS Credentials** you created) from the **Destination** drop-down list.
 - c. Enter the path where the data should be copied to in S3 or ADLS.

For S3, use the following form:

```
s3a://S3_bucket_name/path
```

For ADLS, use the following form:

```
adl://<accountname>.azuredatalakestore.net/<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 or ADLS:

- a. Select the Amazon S3 source (one of the **AWS Credentials** or **Azure Credentials** accounts) 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 or ADLS.

For S3, use the following form:

```
s3a://S3_bucket_name/path_to_metadata_file
```

For ADLS, use the following form:

```
adl://<accountname>.azuredatalakestore.net/<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.
- **Reference Data From Cloud** – Restores only the Hive tables and references the tables on S3 or ADLS as a Hive external table. If you drop a table in Hive, the data remains on S3 or ADLS. 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 or ADLS.

6. Complete the configuration of the Hive/Impala replication schedule by following the steps under [Configuring Replication of Hive/Impala Data](#) on page 98, beginning with step [5.f](#) on page 99

Monitoring the Performance of Hive/Impala Replications



Note: This page contains references to CDH 5 components or features that have been removed from CDH 6. These references are only applicable if you are managing a CDH 5 cluster with Cloudera Manager 6. For more information, see [Deprecated Items](#).

You can monitor the progress of a Hive/Impala replication schedule using performance data that you download as a CSV file from the Cloudera Manager Admin console. This file contains information about the tables and partitions being replicated, the average throughput, and other details that can help diagnose performance issues during Hive/Impala replications. You can view this performance data for running Hive/Impala replication jobs and for completed jobs.

To view the performance data for a *running* Hive/Impala replication schedule:

1. Go to **Backup > Replication Schedules**.
2. Locate the row for the schedule.
3. Click **Performance Reports** and select one of the following options:
 - **HDFS Performance Summary** – downloads a summary performance report of the HDFS phase of the running Hive replication job.
 - **HDFS Performance Full** – downloads a full performance report of the HDFS phase of the running Hive replication job.
 - **Hive Performance** – downloads a report of Hive performance.

Replication Schedules

The screenshot shows the 'Replication Schedules' page in Cloudera Manager. On the left, there are filters for STATUS (Failed, Succeeded, Running, Disabled, Dry-run) and TYPE (HDFS, HDFS-S3, Hive). The main table has columns: ID, Type, Source, Destination, Last Run, and Next Run. The first row is selected, and a context menu is open over the 'Performance Reports' link, showing options: HDFS Performance Summary, HDFS Performance Full, and Hive Performance.

ID	Type	Source	Destination	Last Run	Next Run
5	Hive	HIVE-1 Cluster 1 @ n59	HIVE-1 Cluster 1		None scheduled.

4. To view the data, import the file into a spreadsheet program such as Microsoft Excel.

To view the performance data for a *completed* Hive/Impala replication schedule:

1. Go to **Backup > Replication Schedules**.
2. Locate the schedule and click **Actions > Show History**.
The **Replication History** page for the replication schedule displays.
3. Click **>** to expand the display of the selected schedule.
4. To view performance of the Hive phase, click **Download CSV** next to the **Hive Replication Report** label and select one of the following options:
 - **Results** – download a listing of replicated tables.
 - **Performance** – download a performance report for the Hive replication.

Replication History (Replication Schedules)

Type HIVE Source HIVE-1 (Cluster 1 @ n59u) Destination HIVE-1 (Cluster 1) Next Run None scheduled.

Start Time	Duration	Outcome	Tables	Files Expected	Files Copied	Files Failed	Files Deleted	Files Skipped
December 19, 2016 3:13 PM	4 min	Successful	1	1 (15.4 KiB)	0 (0 B)	0 (0 B)	0	1 (15.4 KiB)

Started At: December 19, 2016 3:13 PM
 Duration: 4 minutes
 Command Details: [View](#)
 Diagnostics: [Collect Diagnostic Data](#)

Message: 1 tables copied.

Download CSV options: HDFS Replication Report, Hive Replication Report, Results, Performance

Display 10 Per Page | << < 1 - 1 >



Note: The option to download the HDFS Replication Report might not appear if the HDFS phase of the replication skipped all HDFS files because they have not changed, or if the **Replicate HDFS Files** option (located on the **Advanced** tab when creating Hive/Impala replication schedules) is not selected.

See [Table 5: Hive Performance Report Columns](#) on page 111 for a description of the data in the HDFS performance reports.

5. To view performance of the HDFS phase, click **Download CSV** next to the **HDFS Replication Report** label and select one of the following options:

- **Listing** – a list of files and directories copied during the replication job.
- **Status** - full status report of files where the status of the replication is one of the following:
 - **ERROR** – An error occurred and the file was not copied.
 - **DELETED** – A deleted file.
 - **SKIPPED** – A file where the replication was skipped because it was up-to-date.
- **Error Status Only** – full status report, filtered to show files with errors only.
- **Deleted Status Only** – full status report, filtered to show deleted files only.
- **Skipped Status Only** – full status report, filtered to show skipped files only.
- **Performance** – summary performance report.
- **Full Performance** – full performance report.

See [Table 1](#) for a description of the data in the HDFS performance reports.

Replication History (Replication Schedules)

Type HIVE Source HIVE-1 (Cluster 1 @ n59u) Destination HIVE-1 (Cluster 1) Next Run None scheduled.

Start Time	Duration	Outcome	Tables	Files Expected	Files Copied	Files Failed	Files Deleted	Files Skipped
December 19, 2016 3:13 PM	4 min	Successful	1	1 (15.4 KiB)	0 (0 B)	0 (0 B)	0	1 (15.4 KiB)

Started At: December 19, 2016 3:13 PM
 Duration: 4 minutes
 Command Details: [View](#)
 Diagnostics: [Collect Diagnostic Data](#)

Message: 1 tables copied.

Download CSV options: HDFS Replication Report, Hive Replication Report, Listing, Status, Error Status Only, Deleted Status Only, Skipped Status Only, Performance, Full Performance

Display 10 Per Page | << < 1 - 1 >

6. To view the data, import the file into a spreadsheet program such as Microsoft Excel.

The performance data is collected every two minutes. Therefore, no data is available during the initial execution of a replication job because not enough samples are available to estimate throughput and other reported data.

The data returned by the CSV files downloaded from the Cloudera Manager Admin console has the following structure:

Table 5: Hive Performance Report Columns

Hive Performance Data Columns	Description
Timestamp	Time when the performance data was collected
Host	Name of the host where the YARN or MapReduce job was running.
DbName	Name of the database.
TableName	Name of the table.
TotalElapsedTimeSecs	Number of seconds elapsed from the start of the copy operation.
TotalTableCount	Total number of tables to be copied. The value of the column will be -1 for replications where Cloudera Manager cannot determine the number of tables being changed.
TotalPartitionCount	Total number of partitions to be copied. If the source cluster is running Cloudera Manager 5.9 or lower, this column contains a value of -1 because older releases do not report this information.
DbCount	Current number of databases copied.
DbErrorCount	Number of failed database copy operations.
TableCount	Total number of tables (for all databases) copied so far.
CurrentTableCount	Total number of tables copied for current database.
TableErrorCount	Total number of failed table copy operations.
PartitionCount	Total number of partitions copied so far (for all tables).
CurrPartitionCount	Total number of partitions copied for the current table.
PartitionSkippedCount	Number of partitions skipped because they were copied in the previous run of the replication job.
IndexCount	Total number of index files copied (for all databases).
CurrIndexCount	Total number of index files copied for the current database.
IndexSkippedCount	Number of Index files skipped because they were not altered. Due to a bug in Hive, this value is always zero.
HiveFunctionCount	Number of Hive functions copied.
ImpalaObjectCount	Number of Impala objects copied.

A sample CSV file, as presented in Excel, is shown here:

Timestamp	Host	DbName	TableName	TotalElapsedTimeSecs	TotalTableCount	TotalPartitionCount	DbCount	DbErrorCount	TableCount	CurrentTableCount	TableErrorCount	PartitionCount	CurrPartitionCount	PartitionSkip	IndexCount	CurrIndexCount	IndexSkippedCount	HiveFunctionCount	ImpalaObjCount
22:16:0	TargetHost-3.m.default	null	null	0	4	-1	1	0	0	0	0	0	0	0	0	0	0	0	0
22:17:6	TargetHost-3.m.null	null	null	1	4	-1	1	0	4	4	0	4	4	0	0	0	0	0	0

Note the following limitations and known issues:

- If you click the CSV download too soon after the replication job starts, Cloudera Manager returns an empty file or a CSV file that has columns headers only and a message to try later when performance data has actually been collected.

Monitoring the Performance of Hive/Impala Replications

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Timestamp	Host	DbName	TableName	TotalElapsed	TotalTableCc	TotalPartitio	DbCount	DbErrorCour	TableCount	CurrentTable	TableErrorCc	Partition
2	No performance statistics available yet: please try again later.												
3													
4													
5													
6													

- If you employ a proxy user with the form `user@domain`, performance data is not available through the links.
- If the replication job only replicates small files that can be transferred in less than a few minutes, no performance statistics are collected.
- For replication schedules that specify the **Dynamic** Replication Strategy, statistics regarding the last file transferred by a MapReduce job hide previous transfers performed by that MapReduce job.
- Only the last trace of each MapReduce job is reported in the CSV file.

Overview of Apache Hive Security in CDH

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](#), [HiveServer2 Security Configuration](#), and [Using Hive to Run Queries on a Secure HBase Server](#) 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 [Authorization With Apache Sentry](#), [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 117 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.

In CDH 5.5 and later, encryption between HiveServer2 and its clients has been decoupled from Kerberos authentication. (Prior to CDH 5.5, SASL QOP encryption for JDBC client drivers required connections authenticated by Kerberos.) De-coupling the authentication process from the transport-layer encryption process means that HiveServer2 can support two different approaches to encryption between the service and its clients (Beeline, JDBC/ODBC) regardless of whether Kerberos is being used for authentication, specifically:

- [SASL](#)
- [TLS/SSL](#)

Unlike TLS/SSL, SASL QOP encryption does not require certificates and is aimed at protecting core Hadoop RPC communications. However, SASL QOP may have performance issues when handling large amounts of data, so depending on your usage patterns, TLS/SSL may be a better choice. See the following topics for details about configuring HiveServer2 services and clients for TLS/SSL and SASL QOP encryption.

See [Configuring Encrypted Communication Between HiveServer2 and Client Drivers](#) on page 114 for details.

Securing the default Hive database

Hive contains a default database named `default`. Everyone can access the database if you set `sentry.hive.restrict.defaultDB=false` in `sentry-site.xml`. You cannot use the default database and perform basic operations, such as listing database names, if this property is set to true.

Accessing information_schema

To query the `information_schema`, `sentry.hive.restrict.defaultDB` must be set to false in `sentry-site.xml`.

Configuring Encrypted Communication Between HiveServer2 and Client Drivers

In CDH 5.5 and later, encryption between HiveServer2 and its clients has been decoupled from Kerberos authentication. (Prior to CDH 5.5, SASL QOP encryption for JDBC client drivers required connections authenticated by Kerberos.) De-coupling the authentication process from the transport-layer encryption process means that HiveServer2 can support two different approaches to encryption between the service and its clients (Beeline, JDBC/ODBC) regardless of whether Kerberos is being used for authentication, specifically:

- [SASL](#)
- [TLS/SSL](#)

Unlike TLS/SSL, SASL QOP encryption does not require certificates and is aimed at protecting core Hadoop RPC communications. However, SASL QOP may have performance issues when handling large amounts of data, so depending on your usage patterns, TLS/SSL may be a better choice. See the following topics for details about configuring HiveServer2 services and clients for TLS/SSL and SASL QOP encryption.

Configuring TLS/SSL Encryption for HiveServer2

HiveServer2 can be configured to support TLS/SSL connections from JDBC/ODBC clients using the Cloudera Manager Admin Console (for clusters that run in the context of Cloudera Manager Server), or manually using the command line.

Requirements and Assumptions

Whether you use Cloudera Manager Admin Console or manually modify the Hive configuration file for TLS/SSL encryption, the steps assume that the HiveServer2 node in the cluster has the necessary server key, certificate, keystore, and trust store set up on the host system. For details, see any of the following:

- [Encrypting Data in Transit](#)
- [How to Configure TLS Encryption for Cloudera Manager](#)
- [How To Obtain and Deploy Keys and Certificates for TLS/SSL](#)

The configuration paths and filenames shown below assume that hostname variable (`$(hostname -f)-server.jks`) was used with Java keytool commands to create keystore, as shown in this example:

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

See the appropriate [How-To guide from the above list](#) for more information.

Using Cloudera Manager to Enable TLS/SSL

To configure TLS/SSL for Hive in clusters managed by Cloudera Manager:

1. Log in to the Cloudera Manager Admin Console.
2. Select **Clusters > Hive**.
3. Click the **Configuration** tab.
4. Select **Hive (Service-Wide)** for the **Scope** filter.
5. Select **Security** for the **Category** filter. The TLS/SSL configuration options display.
6. Enter values for your cluster as follows:

Property	Description
Enable TLS/SSL for HiveServer2	Click the checkbox to enable encrypted client-server communications between HiveServer2 and its clients using TLS/SSL.
HiveServer2 TLS/SSL Server JKS Keystore File Location	Enter the path to the Java keystore on the host system. For example: <div style="border: 1px dashed #0070C0; padding: 5px; margin: 5px 0; width: fit-content;"> <code>/opt/cloudera/security/pki/server-name-server.jks</code> </div>
HiveServer2 TLS/SSL Server JKS Keystore File Password	Enter the password for the keystore that was passed at the Java keytool command-line when the key and keystore were created. As detailed in How To Obtain and Deploy Keys and Certificates for TLS/SSL , the password for the keystore must be the same as the password for the key.
HiveServer2 TLS/SSL Certificate Trust Store File	Enter the path to the Java trust store on the host system. Cloudera clusters are typically configured to use the alternative trust store, <code>jssecacerts</code> , set up at <code>\$JAVA_HOME/jre/lib/security/jssecacerts</code> .

For example:

Enable TLS/SSL for HiveServer2 HIVE-1 (Service-Wide) ↻
hive.server2.enable.SSL,
hive.server2.use.SSL

HiveServer2 TLS/SSL Server JKS Keystore File Location HIVE-1 (Service-Wide) ↻
hive.server2.keystore.path

HiveServer2 TLS/SSL Server JKS Keystore File Password HIVE-1 (Service-Wide) ↻
hive.server2.keystore.password

HiveServer2 TLS/SSL Certificate Trust Store File HIVE-1 (Service-Wide) ↻

HiveServer2 TLS/SSL Certificate Trust Store Password HIVE-1 (Service-Wide)

The entry field for certificate trust store password has been left empty because the trust store is typically not password protected—it contains no keys, only publicly available certificates that help establish the chain of trust during the TLS/SSL handshake. In addition, reading the trust store does not require the password.

7. Click **Save Changes**.
8. Restart the Hive service.

Client Connections to HiveServer2 Over TLS/SSL

Clients connecting to a HiveServer2 over TLS/SSL must be able to access the trust store on the HiveServer2 host system. The trust store contains intermediate and other certificates that the client uses to establish a chain of trust and verify server certificate authenticity. The trust store is typically not password protected.



Note: The trust store may have been password protected to prevent its contents from being modified. However, password protected trust stores can be read from without using the password.

The client needs the path to the trust store when attempting to connect to HiveServer2 using TLS/SSL. This can be specified using two different approaches, as follows:

Configuring Encrypted Communication Between HiveServer2 and Client Drivers

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

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

or,

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

```
java
-Djavax.net.ssl.trustStore=/usr/java/jdk1.7.0_67-cloudera/jre/lib/security/jssecacerts
\
-Djavax.net.ssl.trustStorePassword=extraneous MyClass \
jdbc:hive2://fqdn.example.com:10000/default;ssl=true
```

Configuring SASL Encryption for HiveServer2

Communications between Hive JDBC or ODBC drivers and HiveServer2 can be encrypted using SASL, a framework for authentication and data security rather than a protocol like TLS/SSL. Support for SASL (Simple Authentication and Security Layer) in HiveServer2 preceded the support for TLS/SSL. SASL offers three different Quality of Protection (QOP) levels as shown in the table:

auth	Default. Authentication only.
auth-int	Authentication with integrity protection. Signed message digests (checksums) verify the integrity of messages sent between client and server.
auth-conf	Authentication with confidentiality (transport-layer encryption). Use this setting for encrypted communications from clients to HiveServer2.

To support encryption for communications between client and server processes, specify the QOP `auth-conf` setting for the SASL QOP property in the HiveServer2 configuration file (`hive-site.xml`). For example,

```
<property>
  <name>hive.server2.thrift.sasl.qop</name>
  <value>auth-conf</value>
</property>
```

Client Connections to HiveServer2 Using SASL

The client connection string must match the parameter value specified for the HiveServer2 configuration. This example shows how to specify encryption for the Beeline client in the JDBC connection URL:

```
beeline> !connect jdbc:hive2://fqdn.example.com:10000/default; \
principal=hive/_HOST@EXAMPLE.COM;sasl.qop=auth-conf
```

The `_HOST` is a wildcard placeholder that gets automatically replaced with the fully qualified domain name (FQDN) of the server running the HiveServer2 daemon process.

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 124 section of this page.

See the sections below for details about the supported statements and privileges:

ALTER DATABASE Statement

Use the `ALTER TABLE` statement to set or transfer ownership of an HMS database in Sentry. Object ownership must be enabled in Sentry to assign ownership to an object. For information on how to enable object ownership and the privileges an object owner has on the object, see [Object Ownership](#).

You can grant the OWNER privilege on a database to a role or a user with the following commands, respectively:

```
ALTER DATABASE <database name> SET OWNER ROLE <role name>
ALTER DATABASE <database name> SET OWNER USER <user name>
```

ALTER TABLE Statement

Use the `ALTER TABLE` statement to set or transfer ownership of an HMS table in Sentry. Object ownership must be enabled in Sentry to assign ownership to an object. For information on how to enable object ownership and the privileges an object owner has on the object, see [Object Ownership](#).

You can grant the OWNER privilege on a table to a role or a user with the following commands, respectively:

```
ALTER TABLE <table name> SET OWNER ROLE <role name>
ALTER TABLE <table name> SET OWNER USER <user name>
```

In Hive, the ALTER TABLE statement also sets the owner of a view. Use the following commands to grant the OWNER privilege on a view:

```
ALTER TABLE <view name> SET OWNER ROLE <role name>
ALTER TABLE <view name> SET OWNER USER <user name>
```

ALTER VIEW Statement

In Impala, use the ALTER VIEW statement to transfer ownership of a view in Sentry. Object ownership must be enabled in Sentry to assign ownership to an object. For information on how to enable object ownership and the privileges an object owner has on the object, see [Object Ownership](#).

You can grant the OWNER privilege on a table to a role or a user with the following commands, respectively:

```
ALTER VIEW <view name> SET OWNER ROLE <role name>
ALTER VIEW <view name> SET OWNER USER <user name>
```

In Hive, use the [ALTER TABLE](#) statement to transfer ownership of a view.

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;
```

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>]
```

For example, you might enter the following statement:

```
GRANT SELECT ON TABLE feathered_dinosaurs TO ROLE archaeopteryx
```

The following table describes the privileges you can grant and the objects that they apply to:

Table 6: Privilege Types

Privilege	Object
ALL	Server, database, table, URI
CREATE	Server, database
INSERT	Server, database, table
REFRESH (Impala only)	Server, database, table
SELECT	Server, database, table, view, column

You can also grant the `SELECT` privilege on a specific column of a table with the following statement:

```
GRANT SELECT <column name> ON TABLE <table name> TO ROLE <role name>;
```

GRANT <Privilege> ON URIs (HDFS and S3A)

You can only grant the `ALL` privilege on a URI. See [Granting Privileges on URIs](#) for more information about using URIs with Sentry.

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 add the WITH GRANT OPTION clause to a GRANT <PRIVILEGE> statement to allow the role to grant and revoke the privilege to and from other roles.

The WITH GRANT OPTION clause uses the following syntax:

```
GRANT  
  <privilege>  
  ON <object type> <object name>  
  TO ROLE <role name>  
  WITH GRANT OPTION
```

When you use the WITH GRANT OPTION clause, the ability to grant and revoke privileges applies to the object container and all its children. For example, if you give GRANT privileges to a role at the database level, that role can grant and revoke privileges to and from the database and all the tables in the database.

Only a role with the GRANT option on a privilege can revoke that privilege from other roles. And you cannot revoke the GRANT privilege from a role without also revoking the privilege. To revoke the GRANT privilege, revoke the privilege that it applies to and then grant that privilege again without the WITH GRANT OPTION clause.

You can use the WITH GRANT OPTION clause with the following privileges:

- ALL
- CREATE
- INSERT
- REFRESH (Impala only)
- SELECT

WITH GRANT OPTION Example

For example, if you grant a role the SELECT privilege with the following statement:

```
GRANT SELECT ON DATABASE coffee_database TO ROLE coffee_bean WITH GRANT OPTION
```

The coffee_bean role can grant SELECT privileges to other roles on the coffee_database and all the tables within that database.

When you revoke a privilege from a role, the GRANT privilege is also revoked from that role. For example, if you revoke SELECT privileges from the coffee_bean role with this command:

```
REVOKE SELECT ON DATABASE coffee_database FROM ROLE coffee_bean
```

The coffee_bean role can no longer grant SELECT privileges on the coffee_database or its tables.

To remove the WITH GRANT OPTION privilege from the coffee_bean role and still allow the role to have SELECT privileges on the coffee_database, you must run these two commands:

```
REVOKE SELECT ON coffee_database FROM ROLE coffee_bean;
GRANT SELECT ON coffee_database TO ROLE coffee_bean;
```

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

- Lists the database(s) for which the current user has database, table, or column-level access:

```
SHOW DATABASES;
```

- Lists the table(s) for which the current user has table or column-level access:

```
SHOW TABLES;
```

- Lists the column(s) to which the current user has `SELECT` access:

```
SHOW COLUMNS (FROM|IN) <table name> [(FROM|IN) <database name>];
```

- Lists all the roles in the system (only for sentry admin users):

```
SHOW ROLES;
```

- Lists all the roles in effect for the current user session:

```
SHOW CURRENT ROLES;
```

- Lists 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.

It lists 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>;
```

- Lists all the grants for a role or user 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>;
SHOW GRANT USER <user name> on <object type> <object name>;
```

- Lists the roles and users that have grants on the Hive object. It does not show inherited grants from a parent object. It only shows grants that are applied directly to the object. This command is only available for Hive.

```
SHOW GRANT ON <hive object>;
```

- In Hive, this statement lists all the privileges the user has on objects. In Impala, this statement shows the privileges the user has and the privileges the user's roles have on objects.

```
SHOW GRANT USER <user name>;
```

Privileges

Sentry supports the following privilege types:

CREATE

The CREATE privilege allows a user to create databases, tables, and functions. Note that to create a function, the user also must have ALL permissions on the JAR where the function is located, i.e. GRANT ALL ON URI is required.

You can grant the CREATE privilege on a server or database with the following commands, respectively:

```
GRANT CREATE ON SERVER <server name> TO ROLE <role name>
GRANT CREATE ON DATABASE <database name> TO ROLE <role name>
```

For example, you might enter the following command:

```
GRANT CREATE ON SERVER super_cool_server TO ROLE my_favorite_role
```

You can use the GRANT CREATE statement with the WITH GRANT OPTION clause. The WITH GRANT OPTION clause allows the granted role to grant the privilege to other roles on the system. See [GRANT <Privilege> ... WITH GRANT OPTION](#) on page 120 for more information about how to use the clause.

The following table shows the CREATE privilege scope:

Scope	Available Operations
Server	Create databases, tables, views, and functions
Database	Create tables and views in the database
Table	Not allowed

OWNER

The OWNER privilege gives a user or role special privileges on a database, table, or view in HMS. An object can only have one owner at a time. For more information about the OWNER privilege, see [Object Ownership](#).

The owner of an object can execute any action on the object, similar to the ALL privilege. However, the object owner cannot transfer object ownership unless the **ALL privileges with GRANT** option is selected. You can specify the privileges that an object owner has on the object with the **OWNER Privileges for Sentry Policy Database Objects** setting in Cloudera Manager.

The following table shows the OWNER privilege scope:

Scope	Available Operations
Server	Not available.
Database	<p>Any action allowed by the ALL privilege on the database and tables within the database except transferring ownership of the database or tables.</p> <p>WITH GRANT enabled: Allows the user or role to grant and revoke privileges to other roles on the database, tables, and views. The user can also transfer ownership of the database and tables within the database. If ownership is transferred at the database level, ownership of the tables is not transferred; the original owner continues to have the OWNER privilege on the tables.</p>
Table / View	<p>Any action allowed by the ALL privilege on the table except transferring ownership of the table or view.</p> <p>WITH GRANT enabled: Allows the user or role to transfer ownership of the table or view as well as grant and revoke privileges to other roles on the table or view.</p>

For more information about the OWNER privilege, see [Object Ownership](#).

REFRESH (Impala Only)

The REFRESH privilege allows a user to execute commands that update metadata information on Impala databases and tables, such as the REFRESH and INVALIDATE METADATA commands. Keep in mind that metadata invalidation or refresh in Impala is an expensive procedure that can cause performance issues if it is overused.

You can grant the REFRESH privilege on a server, table, or database with the following commands, respectively:

```
GRANT REFRESH ON SERVER <server name> TO ROLE <role name>
GRANT REFRESH ON DATABASE <database name> TO ROLE <role name>
GRANT REFRESH ON TABLE <table name> TO ROLE <role name>
```

You can use the GRANT REFRESH statement with the WITH GRANT OPTION clause. The WITH GRANT OPTION clause allows the granted role to grant the privilege to other roles on the system. See [GRANT <Privilege> ... WITH GRANT OPTION](#) on page 120 for more information about how to use the clause.

The following table shows the REFRESH privilege scope:

Scope	Available Operations
Server	Invalidate the metadata of all tables on the server
Database	Invalidate the metadata of all tables in the database
Table	Invalidate and refresh the table metadata

SELECT

The SELECT privilege allows a user to view table data and metadata. In addition, you can use the SELECT privilege to provide column-level authorization. See [Column-Level Authorization](#) on page 124 below for details.

You can grant the SELECT privilege on a server, table, or database with the following commands, respectively:

```
GRANT SELECT ON SERVER <server name> TO ROLE <role name>
GRANT SELECT ON DATABASE <database name> TO ROLE <role name>
GRANT SELECT ON TABLE <table name> TO ROLE <role name>
```

Scope	Available Operations
Server	View table data and metadata of all tables in all the databases on the server
Database	View table data and metadata of all tables in the database
Table	View table data and metadata
Column	View table data and metadata for the granted column
View	<ul style="list-style-type: none"> In CDH 5.x, column-level permissions with the SELECT privilege are not available for views. In CDH 6.x, column-level permissions with the SELECT privilege are available for views in Hive, but not in Impala.

Column-Level Authorization

Sentry provides column-level authorization with the SELECT privilege. You can grant the SELECT privilege to a role for a subset of columns in a table. If a new column is added to the table, the role will not have the SELECT privilege on that column until it is explicitly granted.

You can grant and revoke the `SELECT` privilege on a set of columns with the following commands, respectively:

```
GRANT SELECT (<column name>) ON TABLE <table name> TO ROLE <role name>;
REVOKE SELECT (<column name>) ON TABLE <table name> FROM ROLE <role name>;
```

Users with column-level authorization can execute the following commands on the columns that they have access to. Note that the commands will only return data and metadata for the columns that the user's role has been granted access to.

- `SELECT <column name> FROM TABLE <table name>;`
- `SELECT COUNT <column name> FROM TABLE <table name>;`
- `SELECT <column name> FROM TABLE <table name> WHERE <column name> <operator> GROUP BY <column name>;`
- `SHOW COLUMNS (FROM|IN) <table name> [(FROM|IN) <database name>];`

As a rule, a user with select access to columns in a table cannot perform table-level operations, however, if a user has `SELECT` access to all the columns in a table, that user can also execute the following command:

```
SELECT * FROM TABLE <table name>;
```

Considerations for Column-Level Authorization

When you implement column-level authorization, consider the following:

- When a user has column-level permissions, it may be confusing that they cannot execute a `select * from <tablename>` statement even though they have select privileges on some of the columns in the table. Instead, the user must explicitly define the columns that they want to query.
- Using views instead of column-level authorization requires additional administration, such as creating the view and administering the Sentry grants. In addition, a new view may be needed for a new role, and third-party applications must use a different view based on the role of the user.
- With HDFS sync enabled, even if a user has been granted access to all columns of a table, the user 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.

Troubleshooting Apache Hive in CDH

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. In addition, problems can also occur if the metastore metadata gets out of synchronization. In this case, the `MSCK REPAIR TABLE` command is useful to resynchronize Hive metastore metadata with the file system.

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:

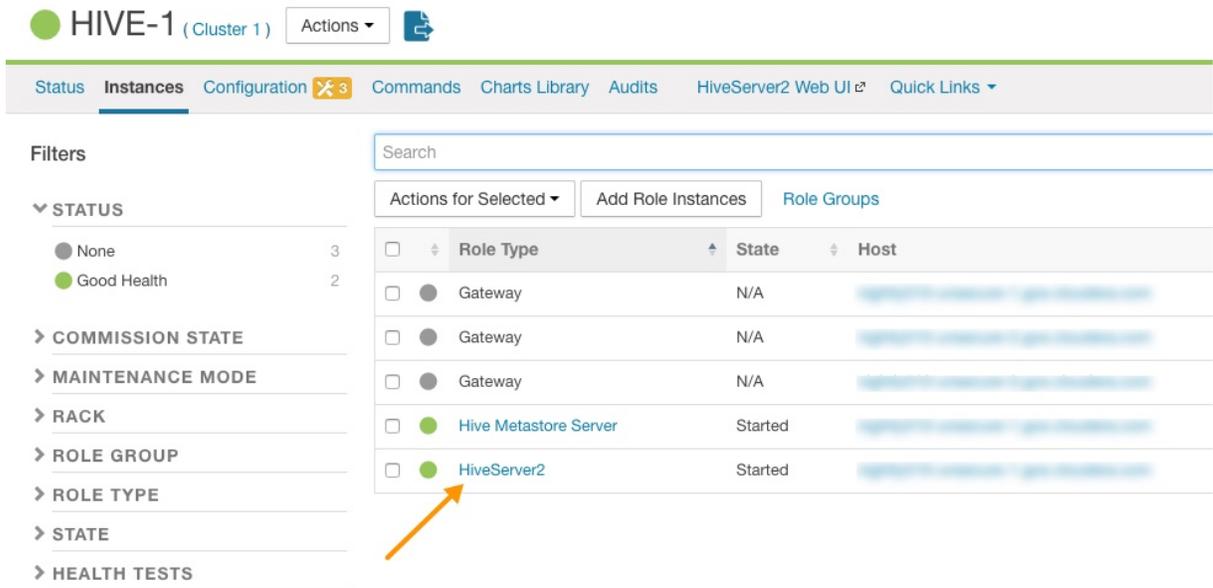


Figure 8: HiveServer2 Link on the Cloudera Manager Instances Page

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.

The screenshot shows the Cloudera Manager interface for HiveServer2. The 'Processes' tab is active, displaying details for the 'hive/hive.sh ["hiveserver2"]' process. The 'Configuration Files' section lists various XML and properties files. The 'Environment Variables' section lists system variables like HIVE_LOG_DIR, HADOOP_CLIENT_OPTS, and SPARK_ON_YARN. At the bottom, the 'Links to full logs' section includes links for 'Stderr', 'Stdout', and 'Role Log Details'. An orange arrow points to the 'Stdout' link.

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.

Best Practices for Using MSCK REPAIR TABLE

Hive stores a list of partitions for each table in its metastore. The MSCK REPAIR TABLE command was designed to bulk-add partitions that already exist on the filesystem but are not present in the metastore. It can be useful if you lose the data in your Hive metastore or if you are working in a cloud environment without a persistent metastore. See [Tuning Apache Hive Performance on the Amazon S3 Filesystem in CDH](#) on page 74 or [Configuring ADLS Gen1 Connectivity](#) on page 56 for more information.

Example: How MSCK REPAIR TABLE Works

The following example illustrates how MSCK REPAIR TABLE works.

1. Create directories and subdirectories on HDFS for the Hive table `employee` and its department partitions:

```
$ sudo -u hive hdfs dfs -mkdir -p /user/hive/dataload/employee/dept=sales
$ sudo -u hive hdfs dfs -mkdir -p /user/hive/dataload/employee/dept=service
$ sudo -u hive hdfs dfs -mkdir -p /user/hive/dataload/employee/dept=finance
```

2. List the directories and subdirectories on HDFS:

```
$ sudo -u hive hdfs hadoop fs -ls -R /user/hive/dataload
drwxr-xr-x - hive hive 0 2017-06-16 17:49 /user/hive/dataload/employee
drwxr-xr-x - hive hive 0 2017-06-16 17:49 /user/hive/dataload/employee/dept=finance
drwxr-xr-x - hive hive 0 2017-06-16 17:47 /user/hive/dataload/employee/dept=sales
drwxr-xr-x - hive hive 0 2017-06-16 17:48 /user/hive/dataload/employee/dept=service
```

3. Use Beeline to create the `employee` table partitioned by `dept`:

```
CREATE EXTERNAL TABLE employee (
  eid int, name string, position string
)
PARTITIONED BY (dept string)
LOCATION '/user/hive/dataload/employee'
;
```

4. Still in Beeline, use the `SHOW PARTITIONS` command on the `employee` table that you just created:

```
SHOW PARTITIONS employee;
```

This command shows none of the partition directories you created in HDFS because the information about these partition directories have not been added to the Hive metastore. Here is the output of `SHOW PARTITIONS` on the `employee` table:

```
+-----+---+
| partition |
+-----+---+
No rows selected (0.118 seconds)
```

5. Use `MSCK REPAIR TABLE` to synchronize the `employee` table with the metastore:

```
MSCK REPAIR TABLE employee;
```

6. Then run the `SHOW PARTITIONS` command again:

```
SHOW PARTITIONS employee;
```

Now this command returns the partitions you created on the HDFS filesystem because the metadata has been added to the Hive metastore:

```
+-----+
| partition |
+-----+
| dept=finance |
| dept=sales   |
| dept=service |
+-----+
3 rows selected (0.089 seconds)
```

Guidelines for Using the MSCK REPAIR TABLE Command

Here are some guidelines for using the MSCK REPAIR TABLE command:

- Running MSCK REPAIR TABLE is very expensive. It consumes a large portion of system resources. Only use it to repair metadata when the metastore has gotten out of sync with the file system. For example, if you transfer data from one HDFS system to another, use MSCK REPAIR TABLE to make the Hive metastore aware of the partitions on the new HDFS. For routine partition creation, use the ALTER TABLE ... ADD PARTITION statement.
- A good use of MSCK REPAIR TABLE is to repair metastore metadata after you move your data files to cloud storage, such as Amazon S3. If you are using this scenario, see [Tuning Hive MSCK \(Metastore Check\) Performance on S3](#) on page 80 for information about tuning MSCK REPAIR TABLE command performance in this scenario.
- Run MSCK REPAIR TABLE as a top-level statement only. Do not run it from inside objects such as routines, compound blocks, or prepared statements.

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.

Appendix: Apache License, Version 2.0

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.
```