

cloudera[®]

Cloudera Administration

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 5.4.x
Date: February 3, 2021

Table of Contents

About Cloudera Administration.....	7
---	----------

Managing CDH and Managed Services.....	8
---	----------

Managing CDH and Managed Services Using Cloudera Manager.....	8
<i>Configuration Overview.....</i>	<i>8</i>
<i>Managing Clusters.....</i>	<i>31</i>
<i>Managing Services.....</i>	<i>35</i>
<i>Managing Roles.....</i>	<i>44</i>
<i>Managing Hosts.....</i>	<i>48</i>
Managing CDH from the Command Line.....	64
<i>Starting CDH Services.....</i>	<i>65</i>
<i>Stopping CDH Services Using the Command Line.....</i>	<i>70</i>
<i>Migrating Data between Clusters Using distcp.....</i>	<i>72</i>
Managing Individual Services.....	76
<i>Managing Flume.....</i>	<i>76</i>
<i>Managing the HBase Service.....</i>	<i>79</i>
<i>Managing HDFS.....</i>	<i>125</i>
<i>Managing Hive.....</i>	<i>153</i>
<i>Managing Hue.....</i>	<i>164</i>
<i>Managing Impala.....</i>	<i>184</i>
<i>Managing Isilon.....</i>	<i>194</i>
<i>Managing Key-Value Store Indexer.....</i>	<i>199</i>
<i>Managing MapReduce and YARN.....</i>	<i>200</i>
<i>Managing Oozie.....</i>	<i>207</i>
<i>Managing Solr.....</i>	<i>216</i>
<i>Managing Spark.....</i>	<i>217</i>
<i>Managing the Sqoop 1 Client.....</i>	<i>232</i>
<i>Managing Sqoop 2.....</i>	<i>233</i>
<i>Managing ZooKeeper.....</i>	<i>233</i>
<i>Configuring Services to Use the GPL Extras Parcel.....</i>	<i>234</i>

Resource Management.....	236
---------------------------------	------------

Schedulers.....	236
Cloudera Manager Resource Management Features.....	236
Managing Resources with Cloudera Manager.....	238
<i>Linux Control Groups.....</i>	<i>238</i>

<i>Static Service Pools</i>	241
<i>Dynamic Resource Pools</i>	242
<i>Managing Impala Admission Control</i>	251
<i>Managing the Impala Llama ApplicationMaster</i>	253
<i>Impala Resource Management</i>	254
<i>Admission Control and Query Queuing</i>	254
<i>Integrated Resource Management with YARN</i>	263

Performance Management.....267

<i>Optimizing Performance in CDH</i>	267
<i>Choosing a Data Compression Format</i>	270
<i>Tuning the Solr Server</i>	271
<i>Tuning to Complete During Setup</i>	271
<i>General Tuning</i>	272
<i>Other Resources</i>	277
<i>Tuning Spark Applications</i>	277
<i>Optimizing Spark Performance</i>	277
<i>Tuning YARN</i>	277
<i>Overview</i>	277
<i>Cluster Configuration</i>	281
<i>YARN Configuration</i>	282
<i>MapReduce Configuration</i>	284
<i>Step 7: MapReduce Configuration</i>	284
<i>Step 7A: MapReduce Sanity Checking</i>	284
<i>Configuring Your Cluster In Cloudera Manager</i>	284

High Availability.....286

<i>HDFS High Availability</i>	286
<i>Introduction to HDFS High Availability</i>	286
<i>Configuring Hardware for HDFS HA</i>	287
<i>Enabling HDFS HA</i>	288
<i>Disabling and Redeploying HDFS HA</i>	301
<i>Configuring Other CDH Components to Use HDFS HA</i>	302
<i>Administering an HDFS High Availability Cluster</i>	305
<i>Changing a Nameservice Name for Highly Available HDFS Using Cloudera Manager</i>	308
<i>MapReduce (MRv1) and YARN (MRv2) High Availability</i>	309
<i>YARN (MRv2) ResourceManager High Availability</i>	309
<i>Work Preserving Recovery for YARN Components</i>	316
<i>MapReduce (MRv1) JobTracker High Availability</i>	318
<i>Cloudera Navigator Key Trustee Server High Availability</i>	330
<i>Configuring Key Trustee Server High Availability Using Cloudera Manager</i>	330
<i>Configuring Key Trustee Server High Availability Using the Command Line</i>	331

<i>Recovering a Key Trustee Server</i>	333
Key Trustee KMS High Availability.....	333
High Availability for Other CDH Components.....	334
<i>HBase High Availability</i>	334
<i>Hive Metastore High Availability</i>	338
<i>Hue High Availability</i>	340
<i>Llama High Availability</i>	343
<i>Configuring Oozie for High Availability</i>	345
<i>Search High Availability</i>	345
Configuring Cloudera Manager for High Availability With a Load Balancer.....	347
<i>Introduction to Cloudera Manager Deployment Architecture</i>	347
<i>Prerequisites for Setting up Cloudera Manager High Availability</i>	349
<i>High-Level Steps to Configure Cloudera Manager High Availability</i>	349
<i>Database High Availability Configuration</i>	374
<i>TLS and Kerberos Configuration for Cloudera Manager High Availability</i>	375

Backup and Disaster Recovery.....377

Backup and Disaster Recovery Overview.....	377
Data Replication.....	378
<i>Designating a Replication Source</i>	379
<i>HBase Replication</i>	380
<i>HDFS Replication</i>	386
<i>Hive Replication</i>	388
<i>Impala Metadata Replication</i>	392
<i>Using Snapshots with Replication</i>	392
<i>Enabling Replication Between Clusters in Different Kerberos Realms</i>	393
<i>Replication of Encrypted Data</i>	393
Snapshots.....	394
<i>Cloudera Manager Snapshot Policies</i>	394
<i>Managing HBase Snapshots</i>	396
<i>Managing HDFS Snapshots</i>	407

Cloudera Manager Administration.....410

Managing the Cloudera Manager Server and Agents.....	410
<i>Starting, Stopping, and Restarting the Cloudera Manager Server</i>	410
<i>Configuring Cloudera Manager Server Ports</i>	410
<i>Moving the Cloudera Manager Server to a New Host</i>	410
<i>Starting, Stopping, and Restarting Cloudera Manager Agents</i>	411
<i>Configuring Cloudera Manager Agents</i>	412
<i>Managing Cloudera Manager Server and Agent Logs</i>	415
<i>Changing Hostnames</i>	416
<i>Configuring Network Settings</i>	418
<i>Managing Alerts</i>	419

Managing Licenses.....421
Sending Usage and Diagnostic Data to Cloudera.....426
Exporting and Importing Cloudera Manager Configuration.....429
Other Cloudera Manager Tasks and Settings.....429
Cloudera Management Service.....430

Cloudera Navigator Data Management Component Administration.....436

Cloudera Navigator Audit Server.....436
Cloudera Navigator Metadata Server.....439

Appendix: Apache License, Version 2.0.....446

About Cloudera Administration

This guide describes how to configure and administer a Cloudera deployment. Administrators manage resources, availability, and backup and recovery configurations. In addition, this guide shows how to implement high availability, and discusses integration.

Managing CDH and Managed Services

If you use Cloudera Manager to manage your cluster, configuring and managing your cluster, as well as individual services and hosts, uses a different paradigm than if you use CDH without Cloudera Manager. For this reason, many of these configuration tasks offer two different subtasks, one each for clusters managed by Cloudera Manager and one for clusters which do not use Cloudera Manager. Often, the tasks are not interchangeable. For instance, if you use Cloudera Manager you cannot use standard Hadoop command-line utilities to start and stop services. Instead, you use Cloudera Manager to perform these tasks.

Managing CDH and Managed Services Using Cloudera Manager

You manage CDH and managed services using the [Cloudera Manager Admin Console](#) and [Cloudera Manager API](#).

The following sections focus on the Cloudera Manager Admin Console.

Configuration Overview

When Cloudera Manager configures a service, it allocates **roles** that are required for that service to the hosts in your cluster. The role determines which service daemons run on a host.

For example, for an HDFS service instance, Cloudera Manager configures:

- One host to run the NameNode role.
- One host to run as the secondary NameNode role.
- One host to run the Balancer role.
- Remaining hosts as to run DataNode roles.

A **role group** is a set of configuration properties for a role type, as well as a list of role instances associated with that group. Cloudera Manager automatically creates a default role group named **Role Type Default Group** for each role type.

When you run the installation or upgrade wizard, Cloudera Manager configures the default role groups it adds, and adds any other required role groups for a given role type. For example, a DataNode role on the same host as the NameNode might require a different configuration than DataNode roles running on other hosts. Cloudera Manager creates a separate role group for the DataNode role running on the NameNode host and uses the default configuration for DataNode roles running on other hosts.

Cloudera Manager wizards [autoconfigure](#) role group properties based on the resources available on the hosts. For properties that are not dependent on host resources, Cloudera Manager default values typically align with CDH default values for that configuration. Cloudera Manager deviates when the CDH default is not a recommended configuration or when the default values are illegal.

Cloudera Manager Configuration Layout

After running the Installation wizard, use Cloudera Manager to reconfigure the existing services and add and configure additional hosts and services.

Cloudera Manager configuration screens offer two layout options: new (the default) and classic. You can switch between layouts using the **Switch to XXX layout** link at the top right of the page. Keep the following in mind when you select a layout:


- If you switch to the classic layout, Cloudera Manager preserves that setting when you upgrade to a new version.
- Selections made in one layout are not preserved when you switch.
- Certain features, including controls for configuring Navigator audit events and HDFS log redaction, are supported only in the new layout.

New layout pages contain controls that allow you filter configuration properties based on configuration status, category, and group. For example, to display the JournalNode maximum log size property (JournalNode Max Log Size), click the **CATEGORY > JournalNode** and **GROUP > Logs** filters:

The screenshot shows the HDFS-1 Configuration page for 'HDFS-1 on Cluster 1'. The 'Configuration' tab is active. The 'Selected Filters' section shows 'x JournalNode' and 'x Logs'. The 'Filters' sidebar on the left is expanded to show 'CATEGORY' and 'GROUP' filters. Under 'CATEGORY', 'JournalNode' is selected with a count of 4. Under 'GROUP', 'Logs' is selected with a count of 4. The main configuration area shows several properties for the 'JournalNode Default Group':

- JournalNode Log Directory:** /var/log/hadoop-hdfs
- JournalNode Logging Threshold:** INFO (selected), with other options: TRACE, DEBUG, WARN, ERROR, FATAL.
- JournalNode Max Log Size:** 200 MIB
- JournalNode Maximum Log File Backups:** 10

At the bottom, there is a 'Display' dropdown set to '25' and 'Entries'.

When a configuration property has been set to a value different from the default, a reset to default value icon  displays.

Classic layout pages are organized by role group and categories within the role group. For example, to display the JournalNode maximum log size property (JournalNode Max Log Size), select **JournalNode Default Group > Logs**.

Configuration

Category	Property	Value
<ul style="list-style-type: none"> ▶ Service-Wide ▶ Balancer Default Group ▶ DataNode Default Group ▶ Failover Controller Default Group ▶ Gateway Default Group ▶ HttpFS Default Group ▼ JournalNode Default Group <ul style="list-style-type: none"> Advanced <li style="background-color: #004a7c; color: white;">Logs Monitoring Performance Ports and Addresses Resource Management 	JournalNode Log Directory	/var/log/hadoop-hdfs default value
	JournalNode Logging Threshold	INFO default value
	JournalNode Max Log Size	200 MIB default value
	JournalNode Maximum Log File Backups	10 default value

When a configuration property has been set to a value different from the default, a **Reset to the default value** link displays.

There is no mechanism for resetting to an [autoconfigured](#) value. However, you can use the configuration [history and rollback feature](#) to revert any configuration changes.

Modifying Configuration Properties



Note: As of Cloudera Manager 5.2, a new layout of the pages where you configure Cloudera Manager system properties was introduced. In Cloudera Manager 5.4, this new layout displays by default. This topic discusses how to configure properties using this new layout. The older layout, called the "Classic Layout" is still available. For instructions on using the classic layout, see [Modifying Configuration Properties \(Classic Layout\)](#) on page 14.

To switch between the layouts, click either the **Switch to the new layout** or **Switch to the classic layout** links in the upper-right portion of all configuration pages.

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

When a service is added to Cloudera Manager, either through the installation or upgrade wizard or with the Add Services workflow, Cloudera Manager automatically sets the configuration properties, based on the needs of the service and characteristics of the cluster in which it will run. These configuration properties include both service-wide configuration properties, as well as specific properties for each role type associated with the service, managed through role groups. A **role group** is a set of configuration properties for a role type, as well as a list of role instances associated with that group. Cloudera Manager automatically creates a default role group named **Role Type Default Group** for each role type. See [Role Groups](#) on page 47.

Changing the Configuration of a Service or Role Instance

1. Go to the service status page. (**Cluster** > **service name**)
2. Click the **Configuration** tab.
3. Locate the property you want to edit. You can type all or part of the property name in the [search box](#), or use the filters on the left side of the screen:
 - The **Status** section limits the displayed properties by their status. Possible statuses include:

- Error
 - Warning
 - Edited
 - Non-default
 - Has Overrides
- The **Scope** section of the left hand panel organizes the configuration properties by role types; first those that are **Service-Wide**, followed by various role types within the service. When you select one of these roles, a set of properties whose values are managed by the default role group for the role display. Any additional role groups that apply to the property also appear in this panel and you can modify values for each role group just as you can the default role group.
 - The **Category** section of the left hand panel allows you to limit the displayed properties by category.

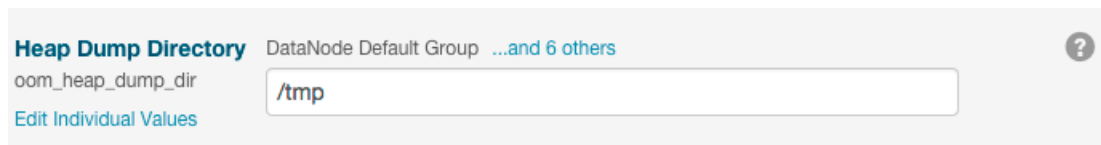
4. Edit the property value.

- To facilitate entering some types of values, you can specify not only the value, but also the units that apply to the value. For example, to enter a setting that specifies bytes per second, you can choose to enter the value in bytes (B), KiBs, MiBs, or GiBs—selected from a drop-down menu that appears when you edit the value.
- If the property allows a list of values, click the **+** icon to the right of the edit field to add an additional field. An example of this is the HDFS DataNode Data Directory property, which can have a comma-delimited list of directories as its value. To remove an item from such a list, click the **-** icon to the right of the field you want to remove.

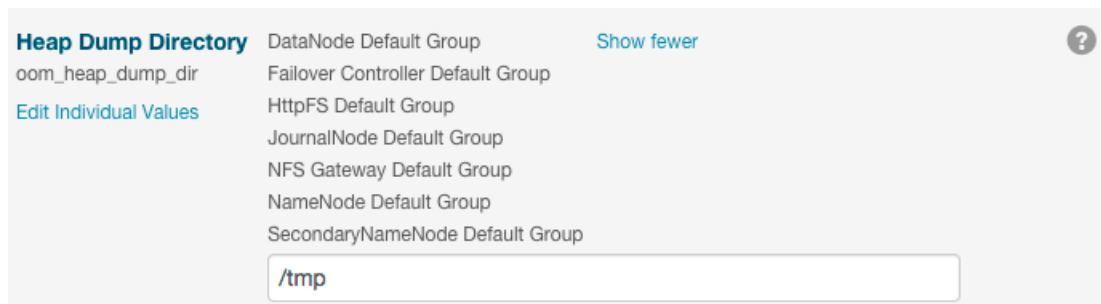
Many configuration properties have different values that are configured by multiple role groups. (See [Role Groups](#) on page 47).

To edit configuration values for multiple role groups:

1. Go to the property, For example, the configuration panel for the **Heap Dump Directory** property displays the DataNode Default Group (a role group), and a link that says **... and 6 others**.



2. Click the **... and 6 others** link to display all of the role groups:



3. Click the **Show fewer** link to collapse the list of role groups.

If you edit the single value for this property, Cloudera Manager applies the value to all role groups. To edit the values for one or more of these role groups individually, click **Edit Individual Values**. Individual fields display where you can edit the values for each role group. For example:

Heap Dump Directory
oom_heap_dump_dir
[Edit Identical Values](#)

DataNode Default Group
/tmp

Failover Controller Default Group
/tmp

HttpFS Default Group
/tmp

JournalNode Default Group
/tmp

NFS Gateway Default Group
/tmp

NameNode Default Group
/tmp

SecondaryNameNode Default Group
/tmp

5. Click **Save Changes** to commit the changes. You can add a note that is included with the change in the Configuration History. This changes the setting for the role group, and applies to all role instances associated with that role group. Depending on the change you made, you may need to restart the service or roles associated with the configuration you just changed. Or, you may need to redeploy your client configuration for the service. You should see a message to that effect at the top of the Configuration page, and services will display an outdated configuration (Restart Needed), (Refresh Needed), or outdated client configuration indicator. Click the indicator to display the [Stale Configurations](#) on page 27 page.

Searching for Properties

You can use the **Search** box to search for properties by name or label. The search also returns properties whose description matches your search term.

Validation of Configuration Properties

Cloudera Manager validates the values you specify for configuration properties. If you specify a value that is outside the recommended range of values or is invalid, Cloudera Manager displays a warning at the top of the **Configuration** tab and in the text box after you click **Save Changes**. The warning is yellow if the value is outside the recommended range of values and red if the value is invalid.

Overriding Configuration Properties


For role types that allow multiple instances, each role instance inherits its configuration properties from its associated role group. While role groups provide a convenient way to provide alternate configuration properties for selected groups of role instances, there may be situations where you want to make a one-off configuration change—for example when a host has malfunctioned and you want to temporarily reconfigure it. In this case, you can override configuration properties for a specific role instance:

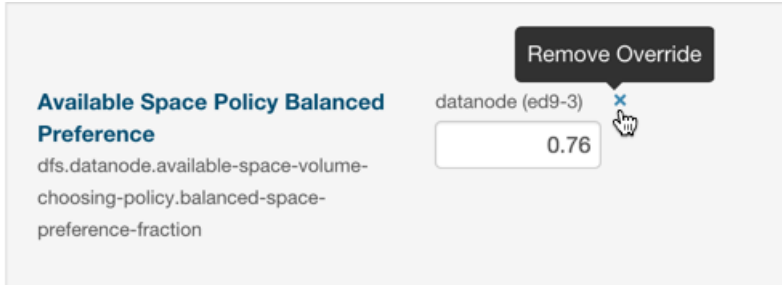
1. Go to the **Status** page for the service whose role you want to change.
2. Click the **Instances** tab.
3. Click the role instance you want to change.
4. Click the **Configuration** tab.
5. Change the configuration values as appropriate.
6. Save your changes.

You will most likely need to restart your service or role to have your configuration changes take effect.


Viewing and Editing Overridden Configuration Properties



To see a list of all role instances that have an override value for a particular configuration setting, go to the Status page for the service and select **Status > Has overrides**. A list of configuration properties where values have been overridden displays. The panel for each configuration property displays the values for each role group or instance. You can edit the value of this property for this instance, or, you can click the

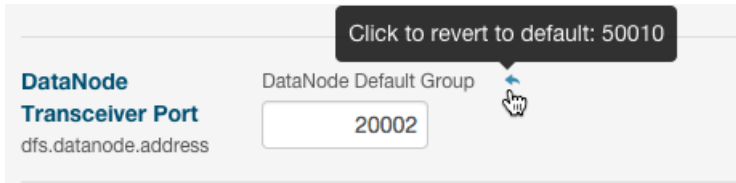
 icon next to an instance name to remove the overridden value.



Resetting Configuration Properties to the Default Value

To reset a property back to its default value, click the  icon. The default value is inserted and the icon turns into an Undo icon

 Explicitly setting a configuration to the same value as its default (inherited value) has the same effect as using the  icon.

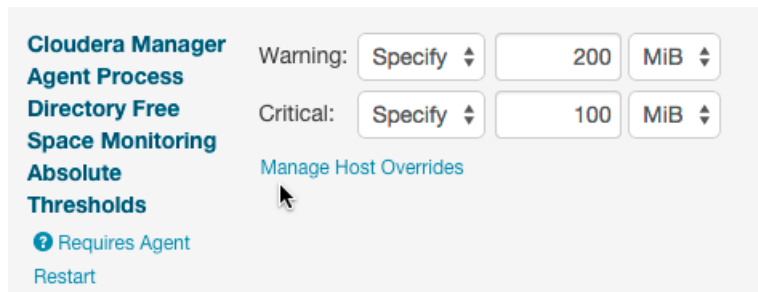


There is no mechanism for resetting to an [autoconfigured](#) value. However, you can use the configuration [history and rollback feature](#) to revert any configuration changes.

Viewing and Editing Host Overrides

You can override the properties of individual hosts in your cluster.

1. Click the **Hosts** tab.
2. Click the **Configuration** tab.
3. Use the Filters or Search box to locate the property that you want to override.
4. Click the **Manage Host Overrides** link.



The **Manage Overrides** dialog box displays.

5. Select one or more hosts to override this property.
6. Click **Update**.

A new entry area displays where you can enter the override values. In the example below, servers `ed9-e.ent.cloudera.com` and `ed9-r.cloudera.com` were selected for overrides. Note that the first set of fields displays the value set for all hosts and the two sets of fields that follow allow you to edit the override values for each specified host.

The screenshot shows the 'Manage Host Overrides' dialog in Cloudera Manager. On the left, there's a sidebar with navigation links: 'Cloudera Manager', 'Agent Process', 'Directory Free', 'Space Monitoring', 'Absolute Thresholds', 'Requires Agent Restart', and 'Edit Identical Values'. The main area is titled 'All Hosts' and shows 'Warning: Specify 200 MiB' and 'Critical: Specify 100 MiB'. Below this, two host-specific sections are visible: 'Cluster 1 > ed9-3.ent.cloudera.com' and 'Cluster 1 > ed9-4.ent.cloudera.com', each with 'Warning: Specify 200 MiB' and 'Critical: Specify 100 MiB'. A 'Manage Host Overrides' link is at the bottom.

To remove the override, click the **X** icon next to the hostname.

To apply the same value to all hosts, click **Edit Identical Values**. Click **Edit Individual Values** to apply different values to selected hosts.

7. If the property indicates **Requires Agent Restart**, restart the agent on the affected hosts.

Restarting Services and Instances after Configuration Changes

If you change the configuration properties after you start a service or instance, you may need to restart the service or instance to have the configuration properties become active. If you change configuration properties at the service level that affect a particular role only (such as all DataNodes but not the NameNodes), you can restart only that role; you do not need to restart the entire service. If you changed the configuration for a particular role instance (such as one of four DataNodes), you may need to restart only that instance.

1. Follow the instructions in [Restarting a Service](#) on page 41 or [Starting, Stopping, and Restarting Role Instances](#) on page 46.
2. If you see a **Finished** status, the service or role instances have restarted.
3. Go to the Home page. The service should show a Status of **Started** for all instances and a health status of **Good**.

For further information, see [Stale Configurations](#) on page 27.

Modifying Configuration Properties (Classic Layout)

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






Note: As of Cloudera Manager version 5.2, a new layout of the pages where you configure Cloudera Manager system properties was introduced. In Cloudera Manager version 5.4, this new layout displays by default. This topic discusses how to configure properties using the older layout, called the "Classic Layout". For instructions on using the new layout, see [Modifying Configuration Properties](#) on page 10.

To switch between the layouts, click either the **Switch to the new layout** or **Switch to the classic layout** links in the upper-right portion of all configuration pages.

When a service is added to Cloudera Manager, either through the installation or upgrade wizard or with the Add Services workflow, Cloudera Manager automatically sets the configuration properties, based on the needs of the service and characteristics of the cluster in which it will run. These configuration properties include both service-wide configuration properties, as well as specific properties for each role type associated with the service, managed through role groups. A **role group** is a set of configuration properties for a role type, as well as a list of role instances associated with that group. Cloudera Manager automatically creates a default role group named **Role Type Default Group** for each role type. See [Role Groups](#) on page 47.

Changing the Configuration of a Service or Role Instance (Classic Layout)

1. Go to the service status page.
2. Click the **Configuration** tab.
3. Under the appropriate role group, select the category for the properties you want to change.
4. To search for a text string (such as "snippet"), in a property, value, or description, enter the text string in the **Search** box at the top of the category list.
5. Moving the cursor over the value cell highlights the cell; click anywhere in the highlighted area to enable editing of the value. Then type the new value in the field provided (or check or uncheck the box, as appropriate).
 - To facilitate entering some types of values, you can specify not only the value, but also the units that apply to the value. For example, to enter a setting that specifies bytes per second, you can choose to enter the value in bytes (B), KiBs, MiBs, or GiBs—selected from a drop-down menu that appears when you edit the value.
 - If the property allows a list of values, click the **+** icon to the right of the edit field to add an additional field. An example of this is the HDFS DataNode Data Directory property, which can have a comma-delimited list of directories as its value. To remove an item from such a list, click the **−** icon to the right of the field you want to remove.
6. Click **Save Changes** to commit the changes. You can add a note that will be included with the change in the Configuration History. This will change the setting for the role group, and will apply to all role instances associated with that role group. Depending on the change you made, you may need to restart the service or roles associated with the configuration you just changed. Or, you may need to redeploy your client configuration for the service. You should see a message to that effect at the top of the Configuration page, and services will display an outdated configuration (Restart Needed), (Refresh Needed), or outdated client configuration  indicator. Click the indicator to display the [Stale Configurations](#) on page 27 page.

Validation of Configuration Properties

Cloudera Manager validates the values you specify for configuration properties. If you specify a value that is outside the recommended range of values or is invalid, Cloudera Manager displays a warning at the top of the **Configuration** tab and in the text box after you click **Save Changes**. The warning is yellow if the value is outside the recommended range of values and red if the value is invalid.

Overriding Configuration Properties

For role types that allow multiple instances, each role instance inherits its configuration properties from its associated role group. While role groups provide a convenient way to provide alternate configuration properties for selected groups of role instances, there may be situations where you want to make a one-off configuration change—for example

when a host has malfunctioned and you want to temporarily reconfigure it. In this case, you can override configuration properties for a specific role instance:

1. Go to the **Status** page for the service whose role you want to change.
2. Click the **Instances** tab.
3. Click the role instance you want to change.
4. Click the **Configuration** tab.
5. Change the configuration values as appropriate.
6. Save your changes.

You will most likely need to restart your service or role to have your configuration changes take effect.

Viewing and Editing Overridden Configuration Properties

To see a list of all role instances that have an override value for a particular configuration setting, go to the entry for the configuration setting in the Status page, expand the **Overridden by *n* instance(s)** link in the value cell for the overridden value.

▼ Overridden by 1 instance(s)
5.0 GiB (DATANODE tcdn5-2.ent.cloudera.com)
[Edit Overrides](#)

To view the override values, and change them if appropriate, click the **Edit Overrides** link. This opens the **Edit Overrides** page, and lists the role instances that have override properties for the selected configuration setting.

Edit Overrides: DataNode Default Group - Reserved Space for Non DFS Use

Reserved space in bytes per volume for non Distributed File System (DFS) use.

Change value of selected instances to:

<input type="checkbox"/>	Role Name	Value
<input type="text" value="Overrides Only"/>		
<input type="checkbox"/>	datanode (tcdn5-2)	5 GiB

On the **Edit Overrides** page, you can do any of the following:

- View the list of role instances that have overridden the value specified in the role group. Use the selections on the drop-down menu below the **Value** column header to view a list of instances that use the inherited value, instances that use an override value, or all instances. This view is especially useful for finding inconsistent properties in a cluster. You can also use the **Host** and **Rack** text boxes to filter the list.
- Change the override value for the role instances to the inherited value from the associated role group. To do so, select the role instances you want to change, choose **Inherited Value** from the drop-down menu next to **Change value of selected instances to** and click **Apply**.
- Change the override value for the role instances to a different value. To do so, select the role instances you want to change, choose **Other** from the drop-down menu next to **Change value of selected instances to**. Enter the new value in the text box and then click **Apply**.

Resetting Configuration Properties to the Default Value

To reset a property back to its default value, click the **Reset to the default value** link below the text box in the value cell. The default value is inserted and both the text box and the Reset link disappear. Explicitly setting a configuration to the same value as its default (inherited value) has the same effect as using the **Reset to the default value** link.

There is no mechanism for resetting to an [autoconfigured](#) value. However, you can use the configuration [history and rollback feature](#) to revert any configuration changes.

Restarting Services and Instances after Configuration Changes

If you change the configuration properties after you start a service or instance, you may need to restart the service or instance to have the configuration properties become active. If you change configuration properties at the service level that affect a particular role only (such as all DataNodes but not the NameNodes), you can restart only that role; you do not need to restart the entire service. If you changed the configuration for a particular role instance (such as one of four DataNodes), you may need to restart only that instance.

1. Follow the instructions in [Restarting a Service](#) on page 41 or [Starting, Stopping, and Restarting Role Instances](#) on page 46.
2. If you see a **Finished** status, the service or role instances have restarted.
3. Go to the Home page. The service should show a Status of **Started** for all instances and a health status of **Good**.

For further information, see [Stale Configurations](#) on page 27.

Autoconfiguration

Cloudera Manager provides several interactive wizards to automate common workflows:

- Installation - used to bootstrap a Cloudera Manager deployment
- Add Cluster - used when adding a new cluster
- Add Service - used when adding a new service
- Upgrade - used when upgrading to a new version of CDH
- Static Service Pools - used when configuring static service pools
- Import MapReduce - used when migrating from MapReduce to YARN

In some of these wizards, Cloudera Manager uses a set of rules to automatically configure certain settings to best suit the characteristics of the deployment. For example, the number of hosts in the deployment drives the memory requirements for certain monitoring daemons: the more hosts, the more memory is needed. Additionally, wizards that are tasked with creating new roles will use a similar set of rules to determine an ideal host placement for those roles.

Scope

The following table shows, for each wizard, the scope of entities it affects during autoconfiguration and role-host placement.

Wizard	Autoconfiguration Scope	Role-Host Placement Scope
Installation	New cluster, Cloudera Management Service	New cluster, Cloudera Management Service
Add Cluster	New cluster	New cluster
Add Service	New service	New service
Upgrade	Cloudera Management Service	Cloudera Management Service
Static Service Pools	Existing cluster	N/A
Import MapReduce	Existing YARN service	N/A

Certain autoconfiguration rules are unscoped, that is, they configure settings belonging to entities that aren't necessarily the entities under the wizard's scope. These exceptions are explicitly listed.

Autoconfiguration

Cloudera Manager employs several different rules to drive automatic configuration, with some variation from wizard to wizard. These rules range from the simple to the complex.

Configuration Scope

One of the points of complexity in autoconfiguration is configuration scope. The configuration hierarchy as it applies to services is as follows: configurations may be modified at the service level (affecting every role in the service), [role group](#) level (affecting every role instance in the group), or role level (affecting one role instance). A configuration found in a lower level takes precedence over a configuration found in a higher level.

With the exception of the Static Service Pools, and the Import MapReduce wizard, all Cloudera Manager wizards follow a basic pattern:

1. Every role in scope is moved into its own, new, role group.
2. This role group is the receptacle for the role's "idealized" configuration. Much of this configuration is driven by properties of the role's host, which can vary from role to role.
3. Once autoconfiguration is complete, new role groups with common configurations are merged.
4. The end result is a smaller set of role groups, each with an "idealized" configuration for some subset of the roles in scope. A subset can have any number of roles; perhaps all of them, perhaps just one, and so on.

The Static Service Pools and Import MapReduce wizards configure role groups directly and do not perform any merging.

Static Service Pools

Certain rules are only invoked in the context of the Static Service Pools wizard. Additionally, the wizard autoconfigures cgroup settings for certain kinds of roles:

- HDFS DataNodes
- HBase RegionServers
- MapReduce TaskTrackers
- YARN NodeManagers
- Impala Daemons
- Solr Servers
- Spark Standalone Workers
- Accumulo Tablet Servers
- Add-on services

YARN

`yarn.nodemanager.resource.cpu-vcores` - For each NodeManager role group, set to number of cores, including hyperthreads, on one NodeManager member's host * service percentage chosen in wizard.

All Services

`Cgroup cpu.shares` - For each role group that supports `cpu.shares`, set to $\max(20, (\text{service percentage chosen in wizard}) * 20)$.

`Cgroup blkio.weight` - For each role group that supports `blkio.weight`, set to $\max(100, (\text{service percentage chosen in wizard}) * 10)$.

Data Directories

Several autoconfiguration rules work with data directories, and there's a common sub-rule used by all such rules to determine, out of all the mountpoints present on a host, which are appropriate for data. The subrule works as follows:

- The initial set of mountpoints for a host includes all those that are disk-backed. Network-backed mountpoints are excluded.
- Mountpoints beginning with `/boot`, `/cdrom`, `/usr`, `/tmp`, `/home`, or `/dev` are excluded.

- Mountpoints beginning with `/media` are excluded, unless the backing device's name contains `/xvd` somewhere in it.
- Mountpoints beginning with `/var` are excluded, unless they are `/var` or `/var/lib`.
- The largest mount point (in terms of total space, not available space) is determined.
- Other mountpoints with less than 1% total space of the largest are excluded.
- Mountpoints beginning with `/var` or equal to `/` are excluded unless they're the largest mount point.
- Remaining mountpoints are sorted lexicographically and retained for future use.

Memory

The rules used to autoconfigure memory reservations are perhaps the most complicated rules employed by Cloudera Manager. When configuring memory, Cloudera Manager must take into consideration which roles are likely to enjoy more memory, and must not over commit hosts if at all possible. To that end, it needs to consider each host as an entire unit, partitioning its available RAM into segments, one segment for each role. To make matters worse, some roles have more than one memory segment. For example, a Solr server has two memory segments: a JVM heap used for most memory allocation, and a JVM direct memory pool used for HDFS block caching. Here is the overall flow during memory autoconfiguration:

1. The set of participants includes every host under scope as well as every {role, memory segment} pair on those hosts. Some roles are under scope while others are not.
2. For each {role, segment} pair where the role is under scope, a rule is run to determine four different values for that pair:
 - Minimum memory configuration. Cloudera Manager must satisfy this minimum, possibly over-committing the host if necessary.
 - Minimum memory consumption. Like the above, but possibly scaled to account for inherent overhead. For example, JVM memory values are multiplied by 1.3 to arrive at their consumption value.
 - Ideal memory configuration. If RAM permits, Cloudera Manager will provide the pair with all of this memory.
 - Ideal memory consumption. Like the above, but scaled if necessary.
3. For each {role, segment} pair where the role is not under scope, a rule is run to determine that pair's existing memory consumption. Cloudera Manager will not configure this segment but will take it into consideration by setting the pair's "minimum" and "ideal" to the memory consumption value.
4. For each host, the following steps are taken:
 - a. 20% of the host's available RAM is subtracted and reserved for the OS.
 - b. `sum(minimum_consumption)` and `sum(ideal_consumption)` are calculated.
 - c. An "availability ratio" is built by comparing the two sums against the host's available RAM.
 - a. If $RAM < \text{sum}(\text{minimum})$ ratio = 0
 - b. If $RAM \geq \text{sum}(\text{ideal})$ ratio = 1
 - c. Otherwise, ratio is computed via: $RAM - \text{sum}(\text{minimum}) / (\text{sum}(\text{ideal}) - \text{sum}(\text{minimum}))$
5. For each {role, segment} pair where the role is under scope, the segment is configured to be $(\text{minimum} + ((\text{ideal} - \text{minimum}) * (\text{host availability ratio})))$. The value is rounded down to the nearest megabyte.
6. The {role, segment} pair is set with the value from the previous step. In the Static Service Pools wizard, the role group is set just once (as opposed to each role).
7. Custom post-configuration rules are run.

Customization rules are applied in steps 2, 3 and 7. In step 2, there's a generic rule for most cases, as well as a series of custom rules for certain {role, segment} pairs. Likewise, there's a generic rule to calculate memory consumption in step 3 as well as some custom consumption functions for certain {role, segment} pairs.

Step 2 Generic Rule Excluding Static Service Pools Wizard

For every {role, segment} pair where the segment defines a default value, the pair's minimum is set to the segment's minimum value (or 0 if undefined), and the ideal is set to the segment's default value.

Step 2 Custom Rules Excluding Static Service Pools Wizard

Managing CDH and Managed Services

HDFS

For the NameNode and Secondary NameNode JVM heaps, the minimum is 50 MB and the ideal is $\max(1 \text{ GB}, \text{sum_over_all}(\text{DataNode mountpoints}' \text{ available space}) / 0.000008)$.

MapReduce

For the JobTracker JVM heap, the minimum is 50 MB and the ideal is $\max(1 \text{ GB}, \text{round}((1 \text{ GB} * 2.3717181092 * \ln(\text{number of TaskTrackers in MapReduce service})) - 2.6019933306))$. If there are ≤ 5 TaskTrackers, the ideal is 1 GB.

For the mapper JVM heaps, the minimum is 1 and the ideal is (number of cores, including hyperthreads, on the TaskTracker host). Note that memory consumption is scaled by `mapred_child_java_opts_max_heap` (the size of a given task's heap).

For the reducer JVM heaps, the minimum is 1 and the ideal is (number of cores, including hyperthreads, on the TaskTracker host) / 2. Note that memory consumption is scaled by `mapred_child_java_opts_max_heap` (the size of a given task's heap).

YARN

For the memory total allowed for containers, the minimum is 1 GB and the ideal is $\min(8 \text{ GB}, (\text{total RAM on NodeManager host}) * 0.8)$.

Hue

With the exception of the Beeswax Server (present only in CDH 4), Hue roles don't have memory limits. Therefore, Cloudera Manager treats them as roles that consume a fixed amount of memory by setting their minimum and ideal consumption values, but not their configuration values. The two consumption values are set to 256 MB.

Impala

With the exception of the Impala Daemon, Impala roles don't have memory limits. Therefore Cloudera Manager treats them as roles that consume a fixed amount of memory by setting their minimum/ideal consumption values, but not their configuration values. The two consumption values are set to 150 MB for the Catalog Server and 64 MB for the StateStore.

For the Impala Daemon memory limit, the minimum is 256 MB and the ideal is $(\text{total RAM on daemon host}) * 0.64$.

Solr

For the Solr Server JVM heap, the minimum is 50 MB and the ideal is $\min(64 \text{ GB}, (\text{total RAM on Solr Server host}) * 0.64) / 2.6$. For the Solr Server JVM direct memory segment, the minimum is 256 MB and the ideal is $\min(64 \text{ GB}, (\text{total RAM on Solr Server host}) * 0.64) / 2$.

Cloudera Management Service

- Alert Publisher JVM heap - treated as if it consumed a fixed amount of memory by setting the minimum/ideal consumption values, but not the configuration values. The two consumption values are set to 256 MB.
- Service and Host Monitor JVM heaps - the minimum is 50 MB and the ideal is either 256 MB (10 or fewer managed hosts), 1 GB (100 or fewer managed hosts), or 2 GB (over 100 managed hosts).
- Event Server, Reports Manager, and Navigator Audit Server JVM heaps - the minimum is 50 MB and the ideal is 1 GB.
- Navigator Metadata Server JVM heap - the minimum is 512 MB and the ideal is 2 GB.
- Service and Host Monitor off-heap memory segments - the minimum is either 768 MB (10 or fewer managed hosts), 2 GB (100 or fewer managed hosts), or 6 GB (over 100 managed hosts). The ideal is always twice the minimum.

Step 2 Generic Rule for Static Service Pools Wizard

For every {role, segment} pair where the segment defines a default value and an autoconfiguration share, the pair's minimum is set to the segment's default value, and the ideal is set to $\min(\text{segment soft max (if exists) or segment max (if exists) or } 2^{63}-1, (\text{total RAM on role's host} * 0.8 / \text{segment scale factor} * \text{service percentage chosen in wizard} * \text{segment autoconfiguration share}))$.

Autoconfiguration shares are defined as follows:

- HBase RegionServer JVM heap: 1
- HDFS DataNode JVM heap: 1 in CDH 4, 0.2 in CDH 5
- HDFS DataNode maximum locked memory: 0.8 (CDH 5 only)
- Solr Server JVM heap: 0.5
- Solr Server JVM direct memory: 0.5
- Spark Standalone Worker JVM heap: 1
- Accumulo Tablet Server JVM heap: 1
- Add-on services: any

Roles not mentioned here do not define autoconfiguration shares and thus aren't affected by this rule.

Additionally, there's a generic rule to handle `cgroup.memory_limit_in_bytes`, which is unused by Cloudera services but is available for add-on services. Its behavior varies depending on whether the role in question has segments or not.

With Segments

The minimum is the $\min(\text{cgroup.memory_limit_in_bytes_min (if exists) or } 0, \text{sum_over_all(segment minimum consumption)})$, and the ideal is the sum of all segment ideal consumptions.

Without Segments

The minimum is `cgroup.memory_limit_in_bytes_min (if exists) or 0`, and the ideal is $(\text{total RAM on role's host} * 0.8 * \text{service percentage chosen in wizard})$.

Step 3 Custom Rules for Static Service Pools Wizard

YARN

For the memory total allowed for containers, the minimum is 1 GB and the ideal is $\min(8 \text{ GB}, (\text{total RAM on NodeManager host}) * 0.8 * \text{service percentage chosen in wizard})$.

Impala

For the Impala Daemon memory limit, the minimum is 256 MB and the ideal is $((\text{total RAM on Daemon host}) * 0.8 * \text{service percentage chosen in wizard})$.

MapReduce

- Mapper JVM heaps - the minimum is 1 and the ideal is $(\text{number of cores, including hyperthreads, on the TaskTracker host} * \text{service percentage chosen in wizard})$. Note that memory consumption is scaled by `mapred_child_java_opts_max_heap` (the size of a given task's heap).
- Reducer JVM heaps - the minimum is 1 and the ideal is $(\text{number of cores, including hyperthreads on the TaskTracker host} * \text{service percentage chosen in wizard}) / 2$. Note that memory consumption is scaled by `mapred_child_java_opts_max_heap` (the size of a given task's heap).

Step 3 Generic Rule

For every {role, segment} pair, the segment's current value is converted into bytes, and then multiplied by the scale factor (1.0 by default, 1.3 for JVM heaps, and freely defined for Custom Service Descriptor services).

Step 3 Custom Rules

Impala

For the Impala Daemon, the memory consumption is 0 if YARN Service for Resource Management is set. If the memory limit is defined but not -1, its value is used verbatim. If it's defined but -1, the consumption is equal to the total RAM on the Daemon host. If it is undefined, the consumption is (total RAM * 0.8).

MapReduce

See [Step 3 Custom Rules for Static Service Pools Wizard](#) on page 21.

Solr

For the Solr Server JVM direct memory segment, the consumption is equal to the value verbatim provided `solr.hdfs.blockcache.enable` and `solr.hdfs.blockcache.direct.memory.allocation` are both true. Otherwise, the consumption is 0.

Step 7 Custom Rules

HDFS

- NameNode JVM heaps are equalized. For every pair of NameNodes in an HDFS service with different heap sizes, the larger heap size is reset to the smaller one.
- JournalNode JVM heaps are equalized. For every pair of JournalNodes in an HDFS service with different heap sizes, the larger heap size is reset to the smaller one.
- NameNode and Secondary NameNode JVM heaps are equalized. For every {NameNode, Secondary NameNode} pair in an HDFS service with different heap sizes, the larger heap size is reset to the smaller one.

HBase

Master JVM heaps are equalized. For every pair of Masters in an HBase service with different heap sizes, the larger heap size is reset to the smaller one.

Impala

If an Impala service has YARN Service for Resource Management set, every Impala Daemon memory limit is set to the value of (`yarn.nodemanager.resource.memory-mb` * 1 GB) if there's a YARN NodeManager co-located with the Impala Daemon.

MapReduce

JobTracker JVM heaps are equalized. For every pair of JobTrackers in an MapReduce service with different heap sizes, the larger heap size is reset to the smaller one.

Oozie

Oozie Server JVM heaps are equalized. For every pair of Oozie Servers in an Oozie service with different heap sizes, the larger heap size is reset to the smaller one.

YARN

ResourceManager JVM heaps are equalized. For every pair of ResourceManagers in a YARN service with different heap sizes, the larger heap size is reset to the smaller one.

ZooKeeper

ZooKeeper Server JVM heaps are equalized. For every pair of servers in a ZooKeeper service with different heap sizes, the larger heap size is reset to the smaller one.

General Rules

HBase

- `hbase.replication` - For each HBase service, set to true if there's a Key-Value Store Indexer service in the cluster. *This rule is unscoped; it can fire even if the HBase service is not under scope.*
- `replication.replicationsource.implementation` - For each HBase service, set to `com.ngdata.sep.impl.SepReplicationSource` if there's a Keystore Indexer service in the cluster. *This rule is unscoped; it can fire even if the HBase service is not under scope.*

HDFS

- `dfs.datanode.du.reserved` - For each DataNode, set to `min((total space of DataNode host largest mountpoint) / 10, 10 GB)`.
- `dfs.namenode.name.dir` - For each NameNode, set to the first two mountpoints on the NameNode host with `/dfs/nn` appended.
- `dfs.namenode.checkpoint.dir` - For each Secondary NameNode, set to the first mountpoint on the Secondary NameNode host with `/dfs/snn` appended.
- `dfs.datanode.data.dir` - For each DataNode, set to all the mountpoints on the host with `/dfs/dn` appended.
- `dfs.journalnode.edits.dir` - For each JournalNode, set to the first mountpoint on the JournalNode host with `/dfs/jn` appended.
- `dfs.datanode.failed.volumes.tolerated` - For each DataNode, set to `(number of mountpoints on DataNode host) / 2`.
- `dfs.namenode.service.handler.count` and `dfs.namenode.handler.count` - For each NameNode, set to `ln(number of DataNodes in this HDFS service) * 20`.
- `dfs.datanode.hdfs-blocks-metadata.enabled` - For each HDFS service, set to true if there's an Impala service in the cluster. *This rule is unscoped; it can fire even if the HDFS service is not under scope.*
- `dfs.client.read.shortcircuit` - For each HDFS service, set to true if there's an Impala service in the cluster. *This rule is unscoped; it can fire even if the HDFS service is not under scope.*
- `dfs.datanode.data.dir.perm` - For each DataNode, set to 755 if there's an Impala service in the cluster and the cluster isn't Kerberized. *This rule is unscoped; it can fire even if the HDFS service is not under scope.*
- `fs.trash.interval` - For each HDFS service, set to 1.

Hue

- **WebHDFS dependency** - For each Hue service, set to either the first HttpFS role in the cluster, or, if there are none, the first NameNode in the cluster.
- **HBase Thrift Server dependency** - For each Hue service in a CDH 4.4 or later cluster, set to the first HBase Thrift Server in the cluster.

Impala

For each Impala service, set **Enable Audit Collection** and **Enable Lineage Collection** to true if there's a Cloudera Management Service with a Navigator Audit Server and Navigator Metadata Server roles. *This rule is unscoped; it can fire even if the Impala service is not under scope.*

MapReduce

- `mapred.local.dir` - For each JobTracker, set to the first mountpoint on the JobTracker host with `/mapred/jt` appended.
- `mapred.local.dir` - For each TaskTracker, set to all the mountpoints on the host with `/mapred/local` appended.
- `mapred.reduce.tasks` - For each MapReduce service, set to `max(1, sum_over_all(TaskTracker number of reduce tasks (determined via mapred.tasktracker.reduce.tasks.maximum for that TaskTracker, which is configured separately)) / 2)`.

Managing CDH and Managed Services

- `mapred.job.tracker.handler.count` - For each JobTracker, set to $\max(10, \ln(\text{number of TaskTrackers in this MapReduce service}) * 20)$.
- `mapred.submit.replication` - If there's an HDFS service in the cluster, for each MapReduce service, set to $\max(1, \sqrt{\text{number of DataNodes in the HDFS service}})$.
- `mapred.tasktracker.instrumentation` - If there's a management service, for each MapReduce service, set to `org.apache.hadoop.mapred.TaskTrackerCmonInst`. *This rule is unscoped; it can fire even if the MapReduce service is not under scope.*

YARN

- `yarn.nodemanager.local-dirs` - For each NodeManager, set to all the mountpoints on the NodeManager host with `/yarn/nm` appended.
- `yarn.nodemanager.resource.cpu-vcores` - For each NodeManager, set to the number of cores (including hyperthreads) on the NodeManager host.
- `mapred.reduce.tasks` - For each YARN service, set to $\max(1, \text{sum_over_all}(\text{NodeManager number of cores, including hyperthreads}) / 2)$.
- `yarn.resourcemanager.nodemanager.heartbeat-interval-ms` - For each NodeManager, set to $\max(100, 10 * (\text{number of NodeManagers in this YARN service}))$.
- `yarn.scheduler.maximum-allocation-vcores` - For each ResourceManager, set to $\max_over_all(\text{NodeManager number of vcores (determined via } \text{yarn.nodemanager.resource.cpu-vcores} \text{ for that NodeManager, which is configured separately)})$.
- `yarn.scheduler.maximum-allocation-mb` - For each ResourceManager, set to $\max_over_all(\text{NodeManager amount of RAM (determined via } \text{yarn.nodemanager.resource.memory-mb} \text{ for that NodeManager, which is configured separately)})$.
- `mapreduce.client.submit.file.replication` - If there's an HDFS service in the cluster, for each YARN service, set to $\max(1, \sqrt{\text{number of DataNodes in the HDFS service}})$.

All Services

If a service dependency is unset, and a service with the desired type exists in the cluster, set the service dependency to the first such target service. Applies to all service dependencies except YARN Service for Resource Management. Applies only to the Installation and Add Cluster wizards.

Role-Host Placement

Cloudera Manager employs the same role-host placement rule regardless of wizard. The set of hosts considered depends on the scope. If the scope is a cluster, all hosts in the cluster are included. If a service, all hosts in the service's cluster are included. If the Cloudera Management Service, all hosts in the deployment are included. The rules are as follows:

1. The hosts are sorted from most to least physical RAM. Ties are broken by sorting on hostname (ascending) followed by host identifier (ascending).
2. The overall number of hosts is used to determine which arrangement to use. These arrangements are hard-coded, each dictating for a given "master" role type, what index (or indexes) into the sorted host list in step 1 to use.
3. Master role types are included based on several factors:
 - Is this role type part of the service (or services) under scope?
 - Does the service already have the right number of instances of this role type?
 - Does the cluster's CDH version support this role type?
 - Does the installed Cloudera Manager license allow for this role type to exist?
4. Master roles are placed on each host using the indexes and the sorted host list. If a host already has a given master role, it is skipped.
5. An HDFS DataNode is placed on every host outside of the arrangement described in step 2, provided HDFS is one of the services under scope.

6. Certain "worker" roles are placed on every host where an HDFS DataNode exists, either because it existed there prior to the wizard, or because it was added in the previous step. The supported worker role types are:
 - MapReduce TaskTrackers
 - YARN NodeManagers
 - HBase RegionServers
 - Impala Daemons
 - Spark Workers
7. Hive gateways are placed on every host, provided a Hive service is under scope and a gateway didn't already exist on a given host.
8. Spark on YARN gateways are placed on every host, provided a Spark on YARN service is under scope and a gateway didn't already exist on a given host.

This rule merely dictates the *default* placement of roles; you are free to modify it before it is applied by the wizard.

Custom Configuration

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

Cloudera Manager exposes properties that allow you to insert custom configuration text into XML configuration, property, and text files, or into an environment. The naming convention for these properties is: **XXX Advanced Configuration Snippet (Safety Valve)** for *YYY* or **XXX YYY Advanced Configuration Snippet (Safety Valve)**, where *XXX* is a service or role and *YYY* is the target.

The values you enter into a configuration snippet must conform to the syntax of the target. For an XML configuration file, the configuration snippet must contain valid XML property definitions. For a properties file, the configuration snippet must contain valid property definitions. Some files simply require a list of host addresses.

The configuration snippet mechanism is intended for use in cases where there is configuration setting that is not exposed as a configuration property in Cloudera Manager. Configuration snippets generally override normal configuration. Contact Cloudera Support if you are required to use a configuration snippet that is not explicitly documented.

Service-wide configuration snippets apply to all roles in the service; a configuration snippet for a role group applies to all instances of the role associated with that role group.

There are configuration snippets for servers and client configurations. In general after changing a server configuration snippet you must [restart](#) the server, and after changing a client configuration snippet you must [redploy the client configuration](#). Sometimes you can refresh instead of restart. In some cases however, you must restart a dependent server after changing a client configuration. For example, changing a MapReduce client configuration marks the dependent Hive server as [stale](#), which must be restarted. The Admin Console displays an indicator when a server must be restarted. In addition, the All Configuration Issues tab on the [Home](#) page lists the actions you must perform to propagate the changes.

Configuration Snippet Types and Syntax

Type	Description	Syntax
Configuration	<p>Set configuration properties in various configuration files; the property name indicates into which configuration file the configuration will be placed. Configuration files have the extension <code>.xml</code> or <code>.conf</code>.</p> <p>For example, there are several configuration snippets for the Hive service. One Hive configuration snippet property is called the HiveServer2 Advanced Configuration Snippet for hive-site.xml;</p>	<pre><property> <name>property_name</name> <value>property_value</value> </property></pre> <p>For example, to specify a MySQL connector library, put this property definition in that configuration snippet:</p> <pre><property> <name>hive.aux.jars.path</name> <value>file:///usr/share/java/mysql-connector-java.jar</value> </property></pre>

Type	Description	Syntax
	<p>configuration you enter here is inserted verbatim into the <code>hive-site.xml</code> file associated with the HiveServer2 role group.</p> <p>To see a list of configuration snippets that apply to a specific configuration file, enter the configuration file name in the Search field in the top navigation bar. For example, searching for <code>mapred-site.xml</code> shows the configuration snippets that have <code>mapred-site.xml</code> in their name.</p>	
Environment	<p>Specify key-value pairs for a service, role, or client that are inserted into the respective environment.</p> <p>One example of using an environment configuration snippet is to add a JAR to a classpath. Place JARs in a custom location such as <code>/opt/myjars</code> and extend the classpath via the appropriate service environment configuration snippet. The value of a JAR property must conform to the syntax supported by its environment. See Setting the class path.</p> <p>Do not place JARs inside locations such as <code>/opt/cloudera</code> or <code>/usr/lib/{hadoop*,hbase*,hive*,etc.}</code> that are managed by Cloudera because they are overwritten at upgrades.</p>	<p><code>key=value</code></p> <p>For example, to add JDBC connectors to a Hive gateway classpath, add</p> <pre>AUX_CLASSPATH=/usr/share/java/mysql-connector-java.jar:\ /usr/share/java/oracle-connector-java.jar</pre> <p>or</p> <pre>AUX_CLASSPATH=/usr/share/java/*</pre> <p>to Gateway Client Advanced Configuration Snippet for <code>hive-env.sh</code>.</p>
Logging	<p>Set properties in a <code>log4j.properties</code> file.</p>	<p><code>key1=value1</code> <code>key2=value2</code></p> <p>For example:</p> <pre>max.log.file.size=200MB max.log.file.backup.index=10</pre>
Metrics	<p>Set properties to configure Hadoop metrics in a <code>hadoop-metrics.properties</code> or <code>hadoop-metrics2.properties</code> file.</p>	<p><code>key1=value1</code> <code>key2=value2</code></p> <p>For example:</p> <pre>*.sink.foo.class=org.apache.hadoop.metrics2.sink.FileSink namenode.sink.foo.filename=/tmp/namenode-metrics.out secondarynamenode.sink.foo.filename=/tmp/secondarynamenode-metrics.out</pre>
White and black lists	<p>Specify a list of host addresses that are allowed or disallowed from accessing a service.</p>	<pre>host1.domain1 host2.domain2</pre>

Setting an Advanced Configuration Snippet

1. Click a service.
2. Click the **Configuration** tab.
3. In the Search box, type `Advanced Configuration Snippet`.

4. Choose a property that contains the string **Advanced Configuration Snippet (Safety Valve)**.
5. Specify the snippet properties.
6. Click **Save Changes** to commit the changes.
7. Restart the service or role or redeploy client configurations as indicated.

Setting Advanced Configuration Snippets for a Cluster or Clusters

1. Do one of the following
 - **specific cluster**
 1. On the Home page, click a cluster name.
 2. Select **Configuration > Advanced Configuration Snippets**.
 - **all clusters**
 1. Select **Configuration > Advanced Configuration Snippets**.
2. Specify the snippet properties.
3. Click **Save Changes** to commit the changes.
4. Restart the service or role or redeploy client configurations as indicated.

Stale Configurations

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

The Stale Configurations page provides differential views of changes made in a cluster. For any configuration change, the page contains entries of all affected attributes. For example, the following File entry shows the change to the file `hdfs-site.xml` when you update the property controlling how much disk space is reserved for non-HDFS use on each DataNode:

```
File: hdfs-site.xml hdfs (3) Show
... .. @@ -91,9 +91,9 @@
91 91 <value>4096</value>
92 92 </property>
93 93 <property>
94 94 <name>dfs.datanode.du.reserved</name>
95 94 - <value>5077964390</value>
95 95 + <value>2147483648</value>
96 96 </property>
97 97 <property>
98 98 <name>dfs.datanode.failed.volumes.tolerated</name>
99 99 <value>0</value>
```

To display the entities affected by a change, click the **Show** button at the right of the entry. The following dialog box shows that three DataNodes were affected by the disk space change:

Entities Affected By This Change ×

Changes From: File: hdfs-site.xml

hdfs 3




datanode (tcdn48-4)

datanode (tcdn48-2)

datanode (tcdn48-3)

Close

Viewing Stale Configurations

To view stale configurations, click the , , or  indicator next to a service on the [Cloudera Manager Admin Console Home Page](#) or on a service status page.

Attribute Categories

The categories of attributes include:

- **Environment** - represents environment variables set for the role. For example, the following entry shows the change to the environment that occurs when you update the heap memory configuration of the SecondaryNameNode.

```
Environment hdfs (1) Show
... .. @@ -2,6 +2,6 @@
2 2 HADOOP_AUDIT_LOGGER=INFO,RFAAUDIT
3 3 HADOOP_LOGFILE=hadoop-cmf-HDFS-1-SECONDARYNAMENODE-tcdn48-1.ent.cloudera.com.log.out
4 4 HADOOP_LOG_DIR=/var/log/hadoop-hdfs
5 5 HADOOP_ROOT_LOGGER=INFO,REA
6 6 -HADOOP_SECONDARYNAMENODE_OPTS=-Xms305135616 -Xmx305135616 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:-CMSConcurrentMTEnabled -XX:CMSInitiatingOccupat
7 7 +HADOOP_SECONDARYNAMENODE_OPTS=-Xms1073741824 -Xmx1073741824 -XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:-CMSConcurrentMTEnabled -XX:CMSInitiatingOccupat
7 7 HADOOP_SECURITY_LOGGER=INFO,RFAS
```

- **Files** - represents configuration files used by the role.
- **Process User & Group** - represents the user and group for the role. Every role type has a configuration to specify the user/group for the process. If you change a value for a user or group on any service's configuration page it will appear in the Stale Configurations page.
- **System Resources** - represents system resources allocated for the role, including ports, directories, and cgroup limits. For example, a change to the port of role instance will appear in the System Resources category.
- **Client Configs Metadata** - represents client configurations.

Filtering Stale Configurations


You filter the entries on the Stale Configurations page by selecting from one of the drop-down lists:

- **Attribute** - you can filter by an attribute category such as All Files or by a specific file such as `topology.map` or `yarn-site.xml`.
- **Service**
- **Role**

After you make a selection, both the page and the drop-down show only entries that match that selection.

To reset the view, click **Remove Filter** or select **All XXX**, where XXX is Files, Services, or Roles, from the drop-down. For example, to see all the files, select **All Files**.

Actions

The Stale Configurations page displays action links. The action depends on what is required to bring the entire cluster's configuration up to date. If you go to the page by clicking a  (Refresh Needed) indicator, the action button will say **Restart Cluster** if *one* of the roles listed on the page need to be restarted.

- **Refresh Cluster** - Runs the [cluster refresh](#) action.
- **Restart Cluster** - Runs the [cluster restart](#) action.
- **Restart Cloudera Management Service** - Runs the [restart Cloudera Management Service](#) action.
- **Deploy Client Configuration** - Runs the [cluster deploy client configurations](#) action.


Client Configuration Files

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

To allow clients to use the HBase, HDFS, Hive, MapReduce, and YARN services, Cloudera Manager creates zip archives of the configuration files containing the service properties. The zip archive is referred to as a **client configuration file**. Each archive contains the set of configuration files needed to access the service: for example, the MapReduce client configuration file contains copies of `core-site.xml`, `hadoop-env.sh`, `hdfs-site.xml`, `log4j.properties` and `mapred-site.xml`.

Client configuration files are generated automatically by Cloudera Manager based on the services and roles you have installed and Cloudera Manager deploys these configurations automatically when you install your cluster, add a service on a host, or add a [gateway role](#) on a host. Specifically, for each host that has a service role instance installed, and for each host that is configured as a gateway role for that service, the deploy function downloads the configuration zip file, unzips it into the appropriate configuration directory, and uses the Linux [alternatives](#) mechanism to set a given, configurable priority level. If you are installing on a system that happens to have pre-existing alternatives, then it is possible another alternative may have higher priority and will continue to be used. The alternatives priority of the Cloudera Manager client configuration is configurable under the **Gateway** sections of the **Configuration** tab for the appropriate service.

You can also manually distribute client configuration files to the clients of a service.

The main circumstance that may require a redeployment of the client configuration files is when you have modified a configuration. In this case you will typically see a message instructing you to redeploy your client configurations. The affected service(s) will also display a  icon. Click the indicator to display the [Stale Configurations](#) on page 27 page.

How Client Configurations are Deployed

Client configuration files are deployed on any host that is a client for a service—that is, that has a role for the service on that host. This includes roles such as DataNodes, TaskTrackers, RegionServers and so on as well as gateway roles for the service.


If roles for multiple services are running on the same host (for example, a DataNode role and a TaskTracker role on the same host) then the client configurations for both roles are deployed on that host, with the alternatives priority determining which configuration takes precedence.

For example, suppose we have six hosts running roles as follows: host H1: HDFS-NameNode; host H2: MR-JobTracker; host H3: HBase-Master; host H4: MR-TaskTracker, HDFS-DataNode, HBase-RegionServer; host H5: MR-Gateway; host H6: HBase-Gateway. Client configuration files will be deployed on these hosts as follows: host H1: hdfs-clientconfig (only); host H2: mapreduce-clientconfig, host H3: hbase-clientconfig; host H4: hdfs-clientconfig, mapreduce-clientconfig, hbase-clientconfig; host H5: mapreduce-clientconfig; host H6: hbase-clientconfig

If the HDFS NameNode and MapReduce JobTracker were on the same host, then that host would have both hdfs-clientconfig and mapreduce-clientconfig installed.

Downloading Client Configuration Files

1. Follow the appropriate procedure according to your starting point:

Page	Procedure
Home	<ol style="list-style-type: none"> 1. On the Home page, click  to the right of the cluster name and select View Client Configuration URLs. A pop-up window with links to the configuration files for the services you have installed displays. 2. Click a link or save the link URL and download the file using <code>wget</code> or <code>curl</code>.
Service	<ol style="list-style-type: none"> 1. Go to a service whose client configuration you want to download. 2. Select Actions > Download Client Configuration.

Manually Redeploying Client Configuration Files

Although Cloudera Manager will deploy client configuration files automatically in many cases, if you have modified the configurations for a service, you may need to redeploy those configuration files.

If your client configurations were deployed automatically, the command described in this section will attempt to redeploy them as appropriate.



Note: If you are deploying client configurations on a host that has multiple services installed, some of the same configuration files, though with different configurations, will be installed in the `conf` directories for each service. Cloudera Manager uses the `priority` parameter in the `alternatives --install` command to ensure that the correct configuration directory is made active based on the combination of services on that host. The priority order is YARN > MapReduce > HDFS. The priority can be configured under the **Gateway** sections of the **Configuration** tab for the appropriate service.

1. On the Home page, click



to the right of the cluster name and select **Deploy Client Configuration**.

2. Click **Deploy Client Configuration**.

Viewing and Reverting Configuration Changes

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



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Whenever you change and save a set of configuration settings for a service or role instance or a host, Cloudera Manager saves a revision of the previous settings and the name of the user who made the changes. You can then view past revisions of the configuration settings, and, if desired, roll back the settings to a previous state.

Viewing Configuration Changes

1. For a service, role, or host, click the **Configuration** tab.
2. Click the **History and Rollback** button. The most recent revision, currently in effect, is shown under **Current Revision**. Prior revisions are shown under **Past Revisions**.
 - By default, or if you click **Show All**, a list of all revisions is shown. If you are viewing a service or role instance, all service/role group related revisions are shown. If you are viewing a host or all hosts, all host/all hosts related revisions are shown.
 - To list only the configuration revisions that were done in a particular time period, use the Time Range Selector to [select a time range](#). Then, click **Show within the Selected Time Range**.
3. Click the **Details...** link. The Revision Details dialog box displays.

Revision Details Dialog

For a service or role instance, shows the following:

- A brief message describing the context of the changes
- The date/time stamp of the change
- The user who performed the change
- The names of any role groups created
- The names of any role groups deleted

For a host instance, shows just a message, date and time stamp, and the user.

The dialog box contains two tabs:

- **Configuration Values** - displays configuration value changes, where changes are organized under the role group to which they were applied. (For example, if you changed a Service-Wide property, it will affect all role groups for that service). For each modified property, the Value column shows the new value of the property and the previous value.
- **Group Membership** - displays changes to the changed the group membership of a role instance (moved the instance from one group to another). This tab is only shown for service and role configurations.

Reverting Configuration Changes

1. Select the current or past revision to which to roll back.
2. Click the **Details...** link. The Revision Details dialog box displays.
3. Click the **Configuration Values** tab.
4. Click the **Revert Configuration Changes** button. The revert action occurs immediately. You may need to restart the service or the affected roles for the change to take effect.



Important: This feature can only be used to revert changes to configuration values. You cannot use this feature to:

- Revert NameNode high availability. You must perform this action by explicitly [disabling high availability](#).
- Disable [Kerberos security](#).
- Revert role group actions (creating, deleting, or moving membership among groups). You must perform these actions explicitly in the [Role Groups](#) on page 47 feature.

Exporting and Importing Cloudera Manager Configuration

You can use the Cloudera Manager API to programmatically export and import a definition of all the entities in your Cloudera Manager-managed deployment—clusters, service, roles, hosts, users and so on. See the [Cloudera Manager API](#) documentation on how to manage deployments using the [/cm/deployment](#) resource.

Managing Clusters

Cloudera Manager can manage multiple clusters. Once you have successfully installed your first cluster, you can add additional clusters, running the same or a different version of CDH. You can then manage each cluster and its services independently.

On the Home page you can access many cluster-wide actions by selecting



to the right of the cluster name: add a service, start, stop, restart, deploy client configurations, enable Kerberos, and perform cluster refresh, rename, upgrade, and maintenance mode actions.



Note:

Cloudera Manager configuration screens offer two layout options: classic and new. The new layout is the default; however, on each configuration page you can easily switch between layouts using the **Switch to XXX layout** link at the top right of the page. For more information, see [Configuration Overview](#) on page 8.

Adding and Deleting Clusters

Minimum Required Role: [Full Administrator](#)

Cloudera Manager can manage multiple clusters. Furthermore, the clusters do not need to run the same version of CDH; you can manage both CDH 4 and CDH 5 clusters with Cloudera Manager.





Important: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, and adding a new cluster.

For more information, see [Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication](#).

Adding a Cluster

Action	Procedure
New Hosts	<ol style="list-style-type: none"> 1. On the Home page, click  and select Add Cluster. This begins the Installation Wizard, just as if you were installing a cluster for the first time. (See Cloudera Manager Deployment for detailed instructions.) 2. To find new hosts, not currently managed by Cloudera Manager, where you want to install CDH, enter the hostnames or IP addresses, and click Search. Cloudera Manager lists the hosts you can use to configure a new cluster. Managed hosts that already have services installed will not be selectable. 3. Click Continue to install the new cluster. At this point the installation continues through the wizard the same as it did when you installed your first cluster. You will be asked to select the version of CDH to install, which services you want and so on, just as previously. 4. Restart the Reports Manager role.
Managed Hosts	<p>You may have hosts that are already "managed" but are not part of a cluster. You can have managed hosts that are not part of a cluster when you have added hosts to Cloudera Manager either through the Add Host wizard, or by manually installing the Cloudera Manager agent onto hosts where you have not install any other services. This will also be the case if you remove all services from a host so that it no longer is part of a cluster.</p> <ol style="list-style-type: none"> 1. On the Home page, click  and select Add Cluster. This begins the Installation Wizard, just as if you were installing a cluster for the first time. (See Cloudera Manager Deployment for detailed instructions.) 2. To see the list of the currently managed hosts, click the Currently Managed Hosts tab. This tab does not appear if you have no currently managed hosts that are not part of a cluster. 3. To perform the installation, click Continue. Instead of searching for hosts, this will attempt to install onto any hosts managed by Cloudera Manager that are not already part of a cluster. It will proceed with the installation wizard as for a new cluster installation. 4. Restart the Reports Manager role.

Deleting a Cluster

1. [Stop](#) the cluster.
2. On the **Home** page, click



to the right of the cluster name and select **Delete**.

Starting, Stopping, Refreshing, and Restarting a Cluster

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

Starting a Cluster

1. On the Home page, click



to the right of the cluster name and select **Start**.

2. Click **Start** that appears in the next screen to confirm. The **Command Details** window shows the progress of starting services.

When **All services successfully started** appears, the task is complete and you can close the **Command Details** window.



Note: The cluster-level Start action starts only CDH and other product services (Impala, Cloudera Search). It does not start the Cloudera Management Service. You must [start the Cloudera Management Service](#) separately if it is not already running.

Stopping a Cluster

1. On the Home page, click



to the right of the cluster name and select **Stop**.

2. Click **Stop** in the confirmation screen. The **Command Details** window shows the progress of stopping services.

When **All services successfully stopped** appears, the task is complete and you can close the **Command Details** window.



Note: The cluster-level Stop action does not stop the Cloudera Management Service. You must [stop the Cloudera Management Service](#) separately.

Refreshing a Cluster

Runs a cluster refresh action to bring the configuration up to date without restarting all services. For example, certain masters (for example NameNode and ResourceManager) have some configuration files (for example, `fair-scheduler.xml`, `mapred_hosts_allow.txt`, `topology.map`) that can be refreshed. If anything changes in those files then a refresh can be used to update them in the master. Here is a summary of the operations performed in a refresh action:

✓ **Refresh Cluster** Cluster 1 Finished Mar 19, 2014 11:31:55 AM PDT Mar 19, 2014 11:32:09 AM PDT

Successfully refreshed roles in the cluster.

Command Progress

Completed 4 of 4 steps.



- ✓ Run 1 steps in parallel
Successfully refreshed datanode allow/exclude lists.
[Details](#) ↗
- ✓ Run 1 steps in parallel
Successfully refreshed ResourceManager.
[Details](#) ↗
- ✓ Run 3 steps in parallel
Successfully refreshed NodeManager.
[Details](#) ↗
- ✓ Run 3 steps in parallel
Refreshed Impala Daemon's Pools configuration and ACLs successfully.
[Details](#) ↗

To refresh a cluster, in the Home page, click



to the right of the cluster name and select **Refresh Cluster**.

Restarting a Cluster

1. On the Home page, click



to the right of the cluster name and select **Restart**.

2. Click **Restart** that appears in the next screen to confirm. The **Command Details** window shows the progress of stopping services.

When **All services successfully started** appears, the task is complete and you can close the **Command Details** window.

Renaming a Cluster

Minimum Required Role: [Full Administrator](#)

1. On the Home page, click



to the right of the cluster name and select **Rename Cluster**.

2. Type the new cluster name and click **Rename Cluster**.

Cluster-Wide Configuration

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

To make configuration changes that apply to an entire cluster, do one of the following to open the configuration page:

- **all clusters**
 1. Select **Configuration** and then select one of the following classes of properties:
 - Advanced Configuration Snippets
 - Databases

- Disk Space Thresholds
- Local Data Directories
- Local Data Files
- Log Directories
- Navigator Settings
- Non-default Values - properties whose value differs from the default value
- Non-uniform Values - properties whose values are not uniform across the cluster or clusters
- Port Configurations
- Service Dependencies

You can also select **Configuration Issues** to view a list of configuration issues for all clusters.

- **specific cluster**

1. On the **Home** page, click a cluster name.
2. Select **Configuration** and then select one of the classes of properties listed above.

You can also apply the following filters to limit the displayed properties:

- Enter a search term in the **Search** box to search for properties by name or description.
- Expand the **Status** filter to select options that limit the displayed properties to those with errors or warnings, properties that have been edited, properties with non-default values, or properties with overrides. Select **All** to remove any filtering by Status.
- Expand the **Scope** filter to display a list of service types. Expand a service type heading to filter on **Service-Wide** configurations for a specific service instance or select one of the default role groups listed under each service type. Select **All** to remove any filtering by Scope.
- Expand the **Category** filter to filter using a sub-grouping of properties. Select **All** to remove any filtering by Category.

Moving a Host Between Clusters

Minimum Required Role: [Full Administrator](#)

Moving a host between clusters can be accomplished by:

1. Decommissioning the host (see [Decommissioning Role Instances](#) on page 46).
2. Removing all roles from the host (except for the Cloudera Manager management roles). See [Deleting Role Instances](#) on page 47.
3. Deleting the host from the cluster (see [Deleting Hosts](#) on page 62), specifically the section on removing a host from a cluster but leaving it available to Cloudera Manager.
4. Adding the host to the new cluster (see [Adding a Host to the Cluster](#) on page 54).
5. Adding roles to the host (optionally using one of the host templates associated with the new cluster). See [Adding a Role Instance](#) on page 45 and [Host Templates](#) on page 57.

Managing Services

Cloudera Manager service configuration features let you manage the deployment and configuration of CDH and managed services. You can add new services and roles if needed, gracefully start, stop and restart services or roles, and decommission and delete roles or services if necessary. Further, you can modify the configuration properties for services or for individual role instances. If you have a Cloudera Enterprise license, you can view past configuration changes and roll back to a previous revision. You can also generate client configuration files, enabling you to easily distribute them to the users of a service.

The topics in this chapter describe how to configure and use the services on your cluster. Some services have unique configuration requirements or provide unique features: those are covered in [Managing Individual Services](#) on page 76.



Note:

Cloudera Manager configuration screens offer two layout options: classic and new. The new layout is the default; however, on each configuration page you can easily switch between layouts using the **Switch to XXX layout** link at the top right of the page. For more information, see [Configuration Overview](#) on page 8.

Adding a Service

Minimum Required Role: [Full Administrator](#)

After initial installation, you can use the **Add a Service** wizard to add and configure new service instances. For example, you may want to add a service such as Oozie that you did not select in the wizard during the initial installation.



Important: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, and adding a new cluster.

For more information, see [Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication](#).

The binaries for the following services are not packaged in CDH 4 and CDH 5 and must be installed individually before being adding the service:

Service	Installation Documentation
Accumulo	Apache Accumulo Documentation
Impala (not in CDH 4)	Installing Impala
Search (not in CDH 4)	Installing Search
Kafka	Installing Kafka
Key Trustee KMS	Installing Key Trustee KMS

If you do not add the binaries before adding the service, the service will fail to start.

To add a service:

1. On the **Home > Status** tab, click



to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.

2. Click the radio button next to a service and click **Continue**. If you are missing required binaries, a pop-up displays asking if you want to continue with adding the service.
3. Select the radio button next to the services on which the new service should depend. All services must depend on the *same* ZooKeeper service. Click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. Review and modify configuration settings, such as data directory paths and heap sizes and click **Continue**. The service is started.
6. Click **Continue** then click **Finish**. You are returned to the [Home](#) page.
7. Verify the new service is started properly by checking the health status for the new service. If the Health Status is **Good**, then the service started properly.

Comparing Configurations for a Service Between Clusters

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

To compare the configuration settings for a particular service between two different clusters in a Cloudera Manager deployment, perform the following steps:

1. On the Home page, click the name of the service you want to compare, or click the **Clusters** menu and select the name of the service.
2. Click the **Configuration** tab.
3. Click the drop-down menu above the Filters pane, and select from one of the options that begins **Diff with...**:
 - **service on cluster** - For example, HBASE-1 on Cluster 1. This is the default display setting. All properties are displayed for the selected instance of the service.
 - **service on all clusters** - For example, HBase on all clusters. All properties are displayed for all instances of the service.
 - **Diff with service on cluster** - For example, Diff with HBase on Cluster 2. Properties are displayed only if the values for the instance of the service whose page you are on differ from the values for the instance selected in the drop-down menu.
 - **Diff with service on all clusters** - For example, Diff with HBase on all clusters. Properties are displayed if the values for the instance of the service whose page you are on differ from the values for one or more other instances in the Cloudera Manager deployment.

The service's properties will be displayed showing the values for each property for the selected clusters. The filters on the left side can be used to limit the properties displayed.

You can also view property configuration values that differ between clusters across a deployment by selecting **Non-uniform Values** on the **Configuration** tab of the Cloudera Manager Home page. For more information, see [Cluster-Wide Configuration](#) on page 34

Add-on Services

Minimum Required Role: [Full Administrator](#)

Cloudera Manager supports adding new types of services (referred to as an **add-on service**) to Cloudera Manager, allowing such services to leverage Cloudera Manager distribution, configuration, monitoring, resource management, and life-cycle management features. An add-on service can be provided by Cloudera or an independent software vendor (ISV). If you have multiple clusters managed by Cloudera Manager, an add-on service can be deployed on any of the clusters.



Note: If the add-on service is already installed and running on hosts that are not currently being managed by Cloudera Manager, you must first add the hosts to a cluster that's under management. See [Adding a Host to the Cluster](#) on page 54 for details.

Custom Service Descriptor Files

Integrating an add-on service requires a Custom Service Descriptor (CSD) file. A CSD file contains all the configuration needed to describe and manage a new service. A CSD is provided in the form of a JAR file.

Depending on the service, the CSD and associated software may be provided by Cloudera or by an ISV. The integration process assumes that the add-on service software (parcel or package) has been installed and is present on the cluster. The recommended method is for the ISV to provide the software as a parcel, but the actual mechanism for installing the software is up to the ISV. The instructions in [Installing an Add-on Service](#) on page 38 assume that you have obtained the CSD file from the Cloudera repository or from an ISV. It also assumes you have obtained the service software, ideally as a parcel, and have or will install it on your cluster either prior to installing the CSD or as part of the CSD installation process.

Configuring the Location of Custom Service Descriptor Files

The default location for CSD files is `/opt/cloudera/csd`. You can change the location in the Cloudera Manager Admin Console as follows:

1. Select **Administration > Settings**.
2. Click the **Custom Service Descriptors** category.
3. Edit the **Local Descriptor Repository Path** property.
4. Click **Save Changes** to commit the changes.
5. Restart Cloudera Manager Server:

```
sudo service cloudera-scm-server restart
```


Installing an Add-on Service

An ISV may provide its software in the form of a parcel, or they may have a different way of installing their software onto your cluster. If their installation process is not via a parcel, then you should install their software before adding the CSD file. Follow the instructions from the ISV for installing the software, if you have not done so already. If the ISV has provided their software as a parcel, they may also have included the location of their parcel repository in the CSD they have provided. In that case, install the CSD first and then install the parcel.

Installing the Custom Service Descriptor File

1. Acquire the CSD file from Cloudera or an ISV.
2. Log on to the Cloudera Manager Server host, and place the CSD file under the location configured for CSD files.
3. Set the file ownership to `cloudera-scm:cloudera-scm` with permission 644.
4. Restart the Cloudera Manager Server:





```
service cloudera-scm-server restart
```

5. Log into the Cloudera Manager Admin Console and restart the Cloudera Management Service.
 - a. Do one of the following:
 - 1. Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - 2. Select **Actions > Restart**.
 - On the Home page, click  to the right of **Cloudera Management Service** and select **Restart**.

- b. Click **Restart** to confirm. The **Command Details** window shows the progress of stopping and then starting the roles.
- c. When **Command completed with n/n successful subcommands** appears, the task is complete. Click **Close**.

Installing the Parcel

If you have already installed the external software onto your cluster, you can skip these steps and proceed to [Adding an Add-on Service](#) on page 39.

1. Click  in the main navigation bar. If the vendor has included the location of the repository in the CSD, the parcel should already be present and ready for downloading. If the parcel is available, skip to [step 7](#).
2. Use one of the following methods to open the parcel settings page:
 - **Navigation bar**
 1. Click  in the top navigation bar
 2. Click the **Edit Settings** button.
 - **Menu**
 1. Select **Administration > Settings**.
 2. Select **Category Parcels**.
3. In the **Remote Parcel Repository URLs** list, click  to open an additional row.
4. Enter the path to the repository.
5. Click **Save Changes** to commit the changes.
6. Click . The external parcel should appear in the set of parcels available for download.
7. Download, distribute, and activate the parcel. See [Managing Parcels](#).

Adding an Add-on Service

Add the service following the procedure in [Adding a Service](#) on page 36.

Uninstalling an Add-on Service

1. Stop all instances of the service.
2. Delete the service from all clusters. If there are other services that depend on the service you are trying to delete, you must delete those services first.
3. Log on to the Cloudera Manager Server host and remove the CSD file.
4. Restart the Cloudera Manager Server:

```
service cloudera-scm-server restart
```

5. After the server has restarted, log into the Cloudera Manager Admin Console and restart the Cloudera Management Service.
6. Optionally remove the parcel.

Starting, Stopping, and Restarting Services

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

Starting and Stopping Services

It's important to start and stop services that have dependencies in the correct order. For example, because MapReduce and YARN have a dependency on HDFS, you must start HDFS before starting MapReduce or YARN. The Cloudera Management Service and Hue are the only two services on which no other services depend; although you can start and stop them at anytime, their preferred order is shown in the following procedures.

The Cloudera Manager cluster actions start and stop services in the correct order. To start or stop all services in a cluster, follow the instructions in [Starting, Stopping, Refreshing, and Restarting a Cluster](#) on page 32.

Starting a Service on All Hosts

1. On the Home page, click



to the right of the service name and select **Start**.

2. Click **Start** that appears in the next screen to confirm. When you see a **Finished** status, the service has started.

The order in which to start services is:

1. Cloudera Management Service
2. ZooKeeper
3. HDFS
4. Solr
5. Flume
6. HBase
7. Key-Value Store Indexer
8. MapReduce or YARN
9. Hive
- 10 Impala
- 11 Oozie
- 12 Sqoop
- 13 Hue



Note: If you are unable to start the HDFS service, it's possible that one of the roles instances, such as a DataNode, was running on a host that is no longer connected to the Cloudera Manager Server host, perhaps because of a hardware or network failure. If this is the case, the Cloudera Manager Server will be unable to connect to the Cloudera Manager Agent on that disconnected host to start the role instance, which will prevent the HDFS service from starting. To work around this, you can stop all services, abort the pending command to start the role instance on the disconnected host, and then restart all services again without that role instance. For information about aborting a pending command, see [Aborting a Pending Command](#) on page 43.

Stopping a Service on All Hosts

1. On the Home page, click



to the right of the service name and select **Stop**.

2. Click **Stop** that appears in the next screen to confirm. When you see a **Finished** status, the service has stopped.

The order in which to stop services is:

1. Hue
2. Sqoop
3. Oozie
4. Impala
5. Hive
6. MapReduce or YARN
7. Key-Value Store Indexer
8. HBase
9. Flume
- 10 Solr
- 11 HDFS
- 12 ZooKeeper

13 Cloudera Management Service

Restarting a Service

It is sometimes necessary to restart a service, which is essentially a combination of stopping a service and then starting it again. For example, if you change the hostname or port where the Cloudera Manager is running, or you enable TLS security, you must restart the Cloudera Management Service to update the URL to the Server.

1. On the Home page, click



to the right of the service name and select **Restart**.

2. Click **Start** on the next screen to confirm. When you see a **Finished** status, the service has restarted.

To restart all services, use the [restart cluster](#) action.

Rolling Restart

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



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Rolling restart allows you to conditionally restart the role instances of Flume, HBase, HDFS, MapReduce, YARN, and ZooKeeper services to update software or use a new configuration. If the service is not running, rolling restart is not available for that service. You can do a rolling restart of each service individually.

If you have [HDFS high availability](#) enabled, you can also perform a cluster-level rolling restart. At the cluster level, the rolling restart of worker hosts is performed on a host-by-host basis, rather than per service, to avoid all roles for a service potentially being unavailable at the same time. During a cluster restart, in order to avoid having your NameNode (and thus the cluster) being unavailable during the restart, Cloudera Manager will force a failover to the standby NameNode.

[MapReduce \(MRv1\) JobTracker High Availability](#) on page 318 and [YARN \(MRv2\) ResourceManager High Availability](#) on page 309 is *not* required for a cluster-level rolling restart. However, if you have JobTracker or ResourceManager high availability enabled, Cloudera Manager will force a failover to the standby JobTracker or ResourceManager.

Performing a Service or Role Rolling Restart

You can initiate a rolling restart from either the Status page for one of the eligible services, or from the service's Instances page, where you can select individual roles to be restarted.

1. Go to the service you want to restart.
2. Do one of the following:
 - **service** - Select **Actions > Rolling Restart**.
 - **role** -
 1. Click the **Instances** tab.
 2. Select the roles to restart.
 3. Select **Actions for Selected > Rolling Restart**.
3. In the pop-up dialog box, select the options you want:
 - Restart only roles whose configurations are stale
 - Restart only roles that are running outdated software versions
 - Which role types to restart
4. If you select an HDFS, HBase, MapReduce, or YARN service, you can have their worker roles restarted in batches. You can configure:

- How many roles should be included in a batch - Cloudera Manager restarts the worker roles rack-by-rack in alphabetical order, and within each rack, hosts are restarted in alphabetical order. If you are using the default replication factor of 3, Hadoop tries to keep the replicas on at least 2 different racks. So if you have multiple racks, you can use a higher batch size than the default 1. But you should be aware that using too high batch size also means that fewer worker roles are active at any time during the upgrade, so it can cause temporary performance degradation. If you are using a single rack only, you should only restart *one worker node at a time* to ensure data availability during upgrade.
- How long should Cloudera Manager wait before starting the next batch.
- The number of *batch* failures that will cause the entire rolling restart to fail (this is an advanced feature). For example if you have a very large cluster you can use this option to allow failures because if you know that your cluster will be functional even if some worker roles are down.



Note:

- **HDFS** - If you do not have HDFS high availability configured, a warning appears reminding you that the service will become unavailable during the restart while the NameNode is restarted. Services that depend on that HDFS service will also be disrupted. It is recommended that you restart the DataNodes one at a time—one host per batch, which is the default.
- **HBase**
 - Administration operations such as any of the following should not be performed during the rolling restart, to avoid leaving the cluster in an inconsistent state:
 - Split
 - Create, disable, enable, or drop table
 - Metadata changes
 - Create, clone, or restore a snapshot. Snapshots rely on the RegionServers being up; otherwise the snapshot will fail.
 - To increase the speed of a rolling restart of the HBase service, set the **Region Mover Threads** property to a higher value. This increases the number of regions that can be moved in parallel, but places additional strain on the HMaster. In most cases, **Region Mover Threads** should be set to 5 or lower.
- **MapReduce** - If you restart the JobTracker, all current jobs will fail.
- **YARN** - If you restart ResourceManager and RM HA is enabled, current jobs continue running: they do not restart or fail. RM HA is supported for CDH 5.2 and higher.
- **ZooKeeper** and **Flume** - For both ZooKeeper and Flume, the option to restart roles in batches is not available. They are always restarted one by one.

5. Click **Confirm** to start the rolling restart.

Performing a Cluster-Level Rolling Restart

You can perform a cluster-level rolling restart on demand from the Cloudera Manager Admin Console. A cluster-level rolling restart is also performed as the last step in a rolling upgrade when the cluster is configured with HDFS high availability enabled.

1. If you have not already done so, enable high availability. See [HDFS High Availability](#) on page 286 for instructions. You do not need to enable automatic failover for rolling restart to work, though you can enable it if you wish. Automatic failover does not affect the rolling restart operation.
2. For the cluster you want to restart select **Actions > Rolling Restart**.
3. In the pop-up dialog box, select the services you want to restart. Please review the caveats in the preceding section for the services you elect to have restarted. The services that do not support rolling restart will simply be restarted, and will be unavailable during their restart.

4. If you select an HDFS, HBase, or MapReduce service, you can have their worker roles restarted in batches. You can configure:
 - How many roles should be included in a batch - Cloudera Manager restarts the worker roles rack-by-rack in alphabetical order, and within each rack, hosts are restarted in alphabetical order. If you are using the default replication factor of 3, Hadoop tries to keep the replicas on at least 2 different racks. So if you have multiple racks, you can use a higher batch size than the default 1. But you should be aware that using too high batch size also means that fewer worker roles are active at any time during the upgrade, so it can cause temporary performance degradation. If you are using a single rack only, you should only restart *one worker node at a time* to ensure data availability during upgrade.
 - How long should Cloudera Manager wait before starting the next batch.
 - The number of *batch* failures that will cause the entire rolling restart to fail (this is an advanced feature). For example if you have a very large cluster you can use this option to allow failures because if you know that your cluster will be functional even if some worker roles are down.
5. Click **Confirm** to start the rolling restart. While the restart is in progress, the Command Details page shows the steps for stopping and restarting the services.


Aborting a Pending Command

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

Commands will time out if they are unable to complete after a period of time.

If necessary, you can abort a pending command. For example, this may become necessary because of a hardware or network failure where a host running a role instance becomes disconnected from the Cloudera Manager Server host. In this case, the Cloudera Manager Server will be unable to connect to the Cloudera Manager Agent on that disconnected host to start or stop the role instance which will prevent the corresponding service from starting or stopping. To work around this, you can abort the command to start or stop the role instance on the disconnected host, and then you can start or stop the service again.

To abort any pending command:


You can click the indicator  with the blue badge, which shows the number of commands that are currently running in your cluster (if any). This indicator is positioned just to the left of the **Support** link at the right hand side of the navigation bar. Unlike the Commands tab for a role or service, this indicator includes all commands running for all services or roles in the cluster. In the Running Commands window, click **Abort** to abort the pending command. For more information, see [Viewing Running and Recent Commands](#).

To abort a pending command for a service or role:

1. Go to the **Service > Instances** tab for the service where the role instance you want to stop is located. For example, go to the **HDFS Service > Instances** tab if you want to abort a pending command for a DataNode.
2. In the list of instances, click the link for role instance where the command is running (for example, the instance that is located on the disconnected host).
3. Go to the **Commands** tab.
4. Find the command in the list of **Running Commands** and click **Abort Command** to abort the running command.

Deleting Services

Minimum Required Role: [Full Administrator](#)

1. Stop the service. For information on starting and stopping services, see [Starting, Stopping, and Restarting Services](#) on page 39.
2. On the Home page, click  to the right of the service name and select **Delete**.
3. Click **Delete** to confirm the deletion. Deleting a service does *not* clean up the associated [client configurations](#) that have been deployed in the cluster or the user data stored in the cluster. For a given "alternatives path" (for example `/etc/hadoop/conf`) if there exist both "live" client configurations (ones that would be pushed out with deploy

client configurations for active services) and ones that have been "orphaned" client configurations (the service they correspond to has been deleted), the orphaned ones will be removed from the alternatives database. In other words, to trigger cleanup of client configurations associated with a deleted service you must create a service to replace it. To remove user data, see [Remove User Data](#).

Renaming a Service

Minimum Required Role: [Full Administrator](#)

A service is given a name upon installation, and that name is used as an identifier internally. However, Cloudera Manager allows you to provide a display name for a service, and that name will appear in the Cloudera Manager Admin Console instead of the original (internal) name.

1. On the Home page, click



to the right of the service name and select **Rename**.

2. Type the new name.
3. Click **Rename**.

The original service name will still be used internally, and may appear or be required in certain circumstances, such as in log messages or in the API.

The rename action is recorded as an Audit event.

When looking at Audit or Event search results for the renamed service, it is possible that these search results might contain either only the original (internal) name, or both the display name and the original name.

Configuring Maximum File Descriptors

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

You can setting the maximum file descriptor parameter for all daemon roles. When not specified, the role uses whatever value it inherits from supervisor. When specified, configures soft and hard limits to the configured value.

1. Go to a service.
2. Click the **Configuration** tab.
3. In the Search box, type **rlimit_fds**.
4. Set the **Maximum Process File Descriptors** property for one or more roles.
5. Click **Save Changes** to commit the changes.
6. Restart the affected role instances.

Managing Roles

When Cloudera Manager configures a service, it configures hosts in your cluster with one or more functions (called roles in Cloudera Manager) that are required for that service. The role determines which Hadoop daemons run on a given host. For example, when Cloudera Manager configures an HDFS service instance it configures one host to run the NameNode role, another host to run as the Secondary NameNode role, another host to run the Balancer role, and some or all of the remaining hosts to run DataNode roles.

Configuration settings are organized in role groups. A **role group** includes a set of configuration properties for a specific group, as well as a list of role instances associated with that role group. Cloudera Manager automatically creates default role groups.

For role types that allow multiple instances on multiple hosts, such as DataNodes, TaskTrackers, RegionServers (and many others), you can create multiple role groups to allow one set of role instances to use different configuration settings than another set of instances of the same role type. In fact, upon initial cluster setup, if you are installing on identical hosts with limited memory, Cloudera Manager will (typically) automatically create two role groups for each worker role — one group for the role instances on hosts with only other worker roles, and a separate group for the instance running on the host that is also hosting master roles.

The HDFS service is an example of this: Cloudera Manager typically creates one role group (DataNode Default Group) for the DataNode role instances running on the worker hosts, and another group (HDFS-1-DATANODE-1) for the DataNode instance running on the host that is also running the master roles such as the NameNode, JobTracker, HBase Master and so on. Typically the configurations for those two classes of hosts will differ in terms of settings such as memory for JVMs.

Cloudera Manager configuration screens offer two layout options: classic and new. The new layout is the default; however, on each configuration page you can easily switch between layouts using the **Switch to XXX layout** link at the top right of the page. For more information, see [Configuration Overview](#) on page 8.

Gateway Roles

A **gateway** is a special type of role whose sole purpose is to designate a host that should receive a client configuration for a specific service, when the host does not have any roles running on it. Gateway roles enable Cloudera Manager to install and manage client configurations on that host. There is no process associated with a gateway role, and its status will always be Stopped. You can configure gateway roles for HBase, HDFS, Hive, MapReduce, Solr, Spark, Sqoop 1 Client, and YARN.

Role Instances

Adding a Role Instance

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

After creating services, you can add role instances to the services. For example, after initial installation in which you created the HDFS service, you can add a DataNode role instance to a host where one was not previously running. Upon upgrading a cluster to a new version of CDH you might want to create a role instance for a role added in the new version. For example, in CDH 5 Impala has the Impala Llama ApplicationMaster role, which must be added after you upgrade a CDH 4 cluster to CDH 5.

1. Go to the service for which you want to add a role instance. For example, to add a DataNode role instance, go to the HDFS service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. Click **Continue**.
6. In the Review Changes page, review the configuration changes to be applied. Confirm the settings entered for file system paths. The file paths required vary based on the services to be installed. For example, you might confirm

the NameNode Data Directory and the DataNode Data Directory for HDFS. Click **Continue**. The wizard finishes by performing any actions necessary to prepare the cluster for the new role instances. For example, new DataNodes are added to the NameNode `dfs_hosts_allow.txt` file. The new role instance is configured with the default role group for its role type, even if there are multiple role groups for the role type. If you want to use a different role group, follow the instructions in [Managing Role Groups](#) on page 48 for moving role instances to a different role group. The new role instances are not started automatically.

Starting, Stopping, and Restarting Role Instances

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

If the host for the role instance is currently decommissioned, you will not be able to start the role until the host has been recommissioned.

1. Go to the service that contains the role instances to start, stop, or restart.
2. Click the **Instances** tab.
3. Check the checkboxes next to the role instances to start, stop, or restart (such as a DataNode instance).
4. Select **Actions for Selected** > **Start, Stop, or Restart**, and then click **Start, Stop, or Restart** again to start the process. When you see a **Finished** status, the process has finished.

Also see [Rolling Restart](#) on page 41.

Decommissioning Role Instances



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

You can remove a role instance such as a DataNode from a cluster while the cluster is running by decommissioning the role instance. When you decommission a role instance, Cloudera Manager performs a procedure so that you can safely retire a host without losing data. Role decommissioning applies to HDFS DataNode, MapReduce TaskTracker, YARN NodeManager, and HBase RegionServer roles.

You cannot decommission a DataNode or a host with a DataNode if the number of DataNodes equals the replication factor (which by default is three) of any file stored in HDFS. For example, if the replication factor of any file is three, and you have three DataNodes, you cannot decommission a DataNode or a host with a DataNode.

A role will be decommissioned if its host is decommissioned. See [Decommissioning and Recommissioning Hosts](#) on page 60 for more details.

To decommission role instances:

1. If you are decommissioning DataNodes, perform the steps in [Tuning HDFS Prior to Decommissioning DataNodes](#) on page 61.
2. Click the service instance that contains the role instance you want to decommission.
3. Click the **Instances** tab.
4. Check the checkboxes next to the role instances to decommission.
5. Select **Actions for Selected** > **Decommission**, and then click **Decommission** again to start the process. While decommissioning is in progress, the role instance displays the  icon. If one role instance of a service is decommissioned, the DECOMMISSIONED facet displays in the Filters on the Instances page and the  icon displays on the role instance's page. When you see a **Finished** status, the decommissioning process has finished.

Recommissioning Role Instances

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

1. Click the service instance that contains the role instance you want to recommission.
2. Click the **Instances** tab.
3. Check the checkboxes next to the decommissioned role instances to recommission.
4. Select **Actions for Selected** > **Recommission**, and then click **Recommission** again to start the process. When you see a **Finished** status, the recommissioning process has finished.

Deleting Role Instances

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

1. Click the service instance that contains the role instance you want to delete. For example, if you want to delete a DataNode role instance, click an HDFS service instance.
2. Click the **Instances** tab.
3. Check the checkboxes next to the role instances you want to delete.
4. If the role instance is running, select **Actions for Selected** > **Stop** and click **Stop** to confirm the action.
5. Select **Actions for Selected** > **Delete**. Click **Delete** to confirm the deletion.



Note: Deleting a role instance does not clean up the associated client configurations that have been deployed in the cluster.

Configuring Roles to Use a Custom Garbage Collection Parameter

Every Java-based role in Cloudera Manager has a configuration setting called **Java Configuration Options for role** where you can enter command line options. Commonly, garbage collection flags or extra debugging flags would be passed here. To find the appropriate configuration setting, select the service you want to modify in the Cloudera Manager Admin Console, then use the Search box to search for Java Configuration Options.

You can add configuration options for all instances of a given role by making this configuration change at the service level. For example, to modify the setting for all DataNodes, select the HDFS service, then modify the **Java Configuration Options for DataNode** setting.

To modify a configuration option for a given instance of a role, select the service, then select the particular role instance (for example, a specific DataNode). The configuration settings you modify will apply to the selected role instance only.

For detailed instructions see [Modifying Configuration Properties](#) on page 10.

Role Groups

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

A **role group** is a set of configuration properties for a role type, as well as a list of role instances associated with that group. Cloudera Manager automatically creates a default role group named **Role Type Default Group** for each role type. Each role instance can be associated with only a single role group.

Role groups provide two types of properties: those that affect the configuration of the service itself and those that affect monitoring of the service, if applicable (the **Monitoring** subcategory). (Not all services have monitoring properties). For more information about monitoring properties see [Configuring Monitoring Settings](#).

When you run the installation or upgrade wizard, Cloudera Manager configures the default role groups it adds, and adds any other required role groups for a given role type. For example, a DataNode role on the same host as the NameNode might require a different configuration than DataNode roles running on other hosts. Cloudera Manager creates a separate role group for the DataNode role running on the NameNode host and uses the default configuration for DataNode roles running on other hosts.

You can modify the settings of the default role group, or you can create new role groups and associate role instances to whichever role group is most appropriate. This simplifies the management of role configurations when one group of role instances may require different settings than another group of instances of the same role type—for example, due to differences in the hardware the roles run on. You modify the configuration for any of the service's role groups through the Configuration tab for the service. You can also [override](#) the settings inherited from a role group for a role instance.

If there are multiple role groups for a role type, you can move role instances from one group to another. When you move a role instance to a different group, it inherits the configuration settings for its new group.

Creating a Role Group



1. Go to a service status page.

2. Click the **Instances** or **Configuration** tab.
3. Click **Role Groups**.
4. Click **Create new group....**
5. Provide a name for the group.
6. Select the role type for the group. You can select role types that allow multiple instances and that exist for the service you have selected.
7. In the **Copy From** field, select the source of the basic configuration information for the role group:
 - An existing role group of the appropriate type.
 - **None....** The role group is set up with generic default values that are *not* the same as the values Cloudera Manager sets in the default role group, as Cloudera Manager specifically sets the appropriate configuration properties for the services and roles it installs. After you create the group you must [edit the configuration](#) to set missing properties (for example the TaskTracker Local Data Directory List property, which is not populated if you select None) and clear other validation warnings and errors.

Managing Role Groups

You can rename or delete existing role groups, and move roles of the same role type from one group to another.

1. Go to a service status page.
2. Click the **Instances** or **Configuration** tab.
3. Click **Role Groups**.
4. Click the group you want to manage. Role instances assigned to the role group are listed.
5. Perform the appropriate procedure for the action:

Action	Procedure
Rename	<ol style="list-style-type: none">1. Click the role group name, click  next to the name on the right and click Rename.2. Specify the new name and click Rename.
Delete	<p>You cannot delete any of the default groups. The group must first be empty; if you want to delete a group you've created, you must move any role instances to a different role group.</p> <ol style="list-style-type: none">1. Click the role group name.2. Click  next to the role group name on the right, select Delete, and confirm by clicking Delete. Deleting a role group removes it from host templates.
Move	<ol style="list-style-type: none">1. Select the role instance(s) to move.2. Select Actions for Selected > Move To Different Role Group....3. In the pop-up that appears, select the target role group and click Move.

Managing Hosts

Cloudera Manager provides a number of features that let you configure and manage the hosts in your clusters.


Content:

The Status Tab

Viewing All Hosts

To display summary information about all the hosts managed by Cloudera Manager, click **Hosts** in the main navigation bar. The All Hosts page displays with a list of all the hosts managed by Cloudera Manager.

The list of hosts shows the overall status of the Cloudera Manager-managed hosts in your cluster.

- The information provided varies depending on which columns are selected. To change the columns, click the **Columns: *n* Selected** drop-down and select the checkboxes next to the columns to display.
- Clicking the  to the left of the number of roles lists all the role instances running on that host. The balloon annotation that appears when you move the cursor over a link indicates the service instance to which the role belongs.
- Filter the hosts by typing a property value in the Search box or selecting a value from the facets at the left of the page. If the [Configuring Agent Heartbeat and Health Status Options](#) on page 412 are configured as follows:
 - Send Agent heartbeat every *x*
 - Set health status to Concerning if the Agent heartbeats fail *y*
 - Set health status to Bad if the Agent heartbeats fail *z*

The value *v* for the Last Heartbeat facet for a host is computed as follows:

- $v < x * y$ = Good
- $v \geq x * y$ and $v \leq x * z$ = Concerning
- $v \geq x * z$ = Bad

Disks Overview

Click the **Disks Overview** button to display an overview of the status of all disks in the deployment. The statistics exposed match or build on those in `iostat`, and are shown in a series of histograms that by default cover every physical disk in the system.

Adjust the endpoints of the time line to see the statistics for different time periods. Specify a filter in the box to limit the displayed data. For example, to see the disks for a single rack `rack1`, set the filter to: `logicalPartition = false and rackId = "rack1"`. Click a histogram to drill down and identify outliers.

Viewing the Hosts in a Cluster

Do one of the following:

- Select **Clusters > Cluster name > General > Hosts**.
- In the Home screen, click  **Hosts** in a full form cluster table.

The All Hosts page displays with a list of the hosts filtered by the cluster name.

Viewing Individual Hosts

You can view detailed information about an individual host—resources (CPU/memory/storage) used and available, which processes it is running, details about the host agent, and much more—by clicking a host link on the All Hosts page. See [Viewing Host Details](#) on page 50.

The Configuration Tab

The **Configuration** tab lets you set properties related to parcels and to resource management, and also monitoring properties for the hosts under management. The configuration settings you make here will affect all your managed hosts. You can also configure properties for individual hosts from the Host Details page (see [Viewing Host Details](#) on page 50) which will override the global properties set here).

To edit the **Default** configuration properties for hosts:

1. Click the **Configuration** tab.

For more information on making configuration changes, see [Modifying Configuration Properties](#) on page 10.

The Templates Tab

The **Templates** tab lets you create and manage [host templates](#), which provide a way to specify a set of role configurations that should be applied to a host. This greatly simplifies the process of adding new hosts, because it lets you specify the configuration for multiple roles on a host in a single step, and then (optionally) start all those roles.

The Parcels Tab

In the **Parcels** tab you can download, distribute, and activate available [parcels](#) to your cluster. You can use parcels to add new products to your cluster, or to upgrade products you already have installed.

Viewing Host Details

You can view detailed information about each host, including:

- Name, IP address, rack ID
- Health status of the host and last time the Cloudera Manager Agent sent a heartbeat to the Cloudera Manager Server
- Number of cores
- System load averages for the past 1, 5, and 15 minutes
- Memory usage
- File system disks, their mount points, and usage
- Health test results for the host
- Charts showing a variety of metrics and health test results over time.
- Role instances running on the host and their health
- CPU, memory, and disk resources used for each role instance



To view detailed host information:

1. Click the **Hosts** tab.
2. Click the name of one of the hosts. The Status page is displayed for the host you selected.
3. Click tabs to access specific categories of information. Each tab provides various categories of information about the host, its services, components, and configuration.

From the status page you can view details about several categories of information.

Status

The Status page is displayed when a host is initially selected and provides summary information about the status of the selected host. Use this page to gain a general understanding of work being done by the system, the configuration, and health status.

If this host has been decommissioned or is in maintenance mode, you will see the following icon(s) (, ) in the top bar of the page next to the status message.

Details

This panel provides basic system configuration such as the host's IP address, rack, health status summary, and disk and CPU resources. This information summarizes much of the detailed information provided in other panes on this tab. To view details about the Host agent, click the Host Agent link in the Details section.

Health Tests

Cloudera Manager monitors a variety of metrics that are used to indicate whether a host is functioning as expected. The Health Tests panel shows health test results in an expandable/collapsible list, typically with the specific metrics that the test returned. (You can Expand All or Collapse All from the links at the upper right of the Health Tests panel).

- The color of the text (and the background color of the field) for a health test result indicates the status of the results. The tests are sorted by their health status – Good, Concerning, Bad, or Disabled. The list of entries for good and disabled health tests are collapsed by default; however, Bad or Concerning results are shown expanded.
- The text of a health test also acts as a link to further information about the test. Clicking the text will pop up a window with further information, such as the meaning of the test and its possible results, suggestions for actions you can take or how to make configuration changes related to the test. The help text for a health test also provides a link to the relevant monitoring configuration section for the service. See [Configuring Monitoring Settings](#) for more information.

Health History

The Health History provides a record of state transitions of the health tests for the host.

- Click the arrow symbol at the left to view the description of the health test state change.
- Click the **View** link to open a new page that shows the state of the host at the time of the transition. In this view some of the status settings are greyed out, as they reflect a time in the past, not the current status.

File Systems

The File systems panel provides information about disks, their mount points and usage. Use this information to determine if additional disk space is required.

Roles

Use the Roles panel to see the role instances running on the selected host, as well as each instance's status and health. Hosts are configured with one or more role instances, each of which corresponds to a service. The role indicates which daemon runs on the host. Some examples of roles include the NameNode, Secondary NameNode, Balancer, JobTrackers, DataNodes, RegionServers and so on. Typically a host will run multiple roles in support of the various services running in the cluster.

Clicking the role name takes you to the role instance's status page.

You can delete a role from the host from the Instances tab of the Service page for the parent service of the role. You can add a role to a host in the same way. See [Role Instances](#) on page 45.

Charts

Charts are shown for each host instance in your cluster.

See [Viewing Charts for Cluster, Service, Role, and Host Instances](#) for detailed information on the charts that are presented, and the ability to search and display metrics of your choice.

Processes

The Processes page provides information about each of the processes that are currently running on this host. Use this page to access management web UIs, check process status, and access log information.



Note: The Processes page may display exited startup processes. Such processes are cleaned up within a day.

The Processes tab includes a variety of categories of information.

- **Service** - The name of the service. Clicking the service name takes you to the service status page. Using the triangle to the right of the service name, you can directly access the tabs on the role page (such as the Instances, Commands, Configuration, Audits, or Charts Library tabs).
- **Instance** - The role instance on this host that is associated with the service. Clicking the role name takes you to the role instance's status page. Using the triangle to the right of the role name, you can directly access the tabs on the role page (such as the Processes, Commands, Configuration, Audits, or Charts Library tabs) as well as the status page for the parent service of the role.
- **Name** - The process name.

- **Links** - Links to management interfaces for this role instance on this system. These is not available in all cases.
- **Status** - The current status for the process. Statuses include stopped, starting, running, and paused.
- **PID** - The unique process identifier.
- **Uptime** - The length of time this process has been running.
- **Full log file** - A link to the full log (a file external to Cloudera Manager) for this host log entries for this host.
- **Stderr** - A link to the stderr log (a file external to Cloudera Manager) for this host.
- **Stdout** - A link to the stdout log (a file external to Cloudera Manager) for this host.

Resources

The Resources page provides information about the resources (CPU, memory, disk, and ports) used by every service and role instance running on the selected host.

Each entry on this page lists:

- The service name
- The name of the particular instance of this service
- A brief description of the resource
- The amount of the resource being consumed or the settings for the resource

The resource information provided depends on the type of resource:

- **CPU** - An approximate percentage of the CPU resource consumed.
- **Memory** - The number of bytes consumed.
- **Disk** - The disk location where this service stores information.
- **Ports** - The port number being used by the service to establish network connections.

Commands

The Commands page shows you running or recent commands for the host you are viewing. See [Viewing Running and Recent Commands](#) for more information.

Configuration

Minimum Required Role: [Full Administrator](#)

The Configuration page for a host lets you set properties for the selected host. You can set properties in the following categories:

- **Advanced** - Advanced configuration properties. These include the Java Home Directory, which explicitly sets the value of `JAVA_HOME` for all processes. This overrides the auto-detection logic that is normally used.
- **Monitoring** - Monitoring properties for this host. The monitoring settings you make on this page will override the global host monitoring settings you make on the Configuration tab of the Hosts page. You can configure monitoring properties for:
 - health check thresholds
 - the amount of free space on the filesystem containing the Cloudera Manager Agent's log and process directories
 - a variety of conditions related to memory usage and other properties
 - alerts for health check events

For some monitoring properties, you can set thresholds as either a percentage or an absolute value (in bytes).

- **Other** - Other configuration properties.
- **Parcels** - Configuration properties related to parcels. Includes the **Parcel Director** property, the directory that parcels will be installed into on this host. If the `parcel_dir` variable is set in the Agent's `config.ini` file, it will override this value.
- **Resource Management** - Enables resource management using control groups (cgroups).

For more information, see the description for each or property or see [Modifying Configuration Properties](#) on page 10.

Components

The Components page lists every component installed on this host. This may include components that have been installed but have not been added as a service (such as YARN, Flume, or Impala).

This includes the following information:

- **Component** - The name of the component.
- **Version** - The version of CDH from which each component came.
- **Component Version** - The detailed version number for each component.

Audits

The Audits page lets you filter for audit events related to this host. See [Audit Events](#) for more information.

Charts Library

The Charts Library page for a host instance provides charts for all metrics kept for that host instance, organized by category. Each category is collapsible/expandable. See [Viewing Charts for Cluster, Service, Role, and Host Instances](#) for more information.

Using the Host Inspector

Minimum Required Role: [Full Administrator](#)

You can use the host inspector to gather information about hosts that Cloudera Manager is currently managing. You can review this information to better understand system status and troubleshoot any existing issues. For example, you might use this information to investigate potential DNS misconfiguration.

The inspector runs tests to gather information for functional areas including:

- Networking
- System time
- User and group configuration
- HDFS settings
- Component versions

Common cases in which this information is useful include:

- Installing components
- Upgrading components
- Adding hosts to a cluster
- Removing hosts from a cluster

Running the Host Inspector

1. Click the **Hosts** tab.
2. Click **Host Inspector**. Cloudera Manager begins several tasks to inspect the managed hosts.
3. After the inspection completes, click **Download Result Data** or **Show Inspector Results** to review the results.

The results of the inspection displays a list of all the validations and their results, and a summary of all the components installed on your managed hosts.

If the validation process finds problems, the **Validations** section will indicate the problem. In some cases the message may indicate actions you can take to resolve the problem. If an issue exists on multiple hosts, you may be able to view the list of occurrences by clicking a small triangle that appears at the end of the message.

The **Version Summary** section shows all the components that are available from Cloudera, their versions (if known) and the CDH distribution to which they belong (CDH 4 or CDH 5).

If you are running multiple clusters with both CDH 4 and CDH 5, the lists will be organized by distribution (CDH 4 or CDH 5). The hosts running that version are shown at the top of each list.

Viewing Past Host Inspector Results

You can view the results of a past host inspection by looking for the Host Inspector command using the **Recent Commands** feature.

1. Click the Running Commands indicator (🔄) just to the left of the Search box at the right hand side of the navigation bar.
2. Click the **Recent Commands** button.
3. If the command is too far in the past, you can use the Time Range Selector to move the time range back to cover the time period you want.
4. When you find the Host Inspector command, click its name to display its subcommands.
5. Click the **Show Inspector Results** button to view the report.

See [Viewing Running and Recent Commands](#) for more information about viewing past command activity.

Adding a Host to the Cluster

Minimum Required Role: [Full Administrator](#)

You can add one or more hosts to your Hadoop cluster using the Add Hosts wizard, which will install the Oracle JDK, CDH, Impala (optional) and the Cloudera Manager Agent packages. After these packages are installed and the Cloudera Manager Agent is started, the Agent will connect to the Cloudera Manager Server and you will then be able to use the Cloudera Manager Admin Console to manage and monitor CDH on the new host.

The Add Hosts wizard does not create roles on the new host; once you have successfully added the host(s) you can either add roles, one service at a time, or apply a host template, which can define role configurations for multiple roles.



Important: As of February 1, 2021, all downloads of CDH and Cloudera Manager require a username and password and use a modified URL. You must use the modified URL, including the username and password when downloading the repository contents described below. You may need to upgrade Cloudera Manager to a newer version that uses the modified URLs.

This can affect new installations, upgrades, adding new hosts to a cluster, and adding a new cluster.

For more information, see [Updating an existing CDH/Cloudera Manager deployment to access downloads with authentication](#).



Important:

- All hosts in a single cluster must be running the same version of CDH, for example CDH 4.5 or CDH 5.0.
- When you install the new hosts on your system, you must install the same version of CDH to enable the new host to work with the other hosts in the cluster. The installation wizard lets you select the version of CDH you want to install, and you can choose a custom repository to ensure that the version you install matches the version on your other hosts.
- If you are managing multiple clusters, be sure to select the version of CDH that matches the version in use on the cluster where you plan to add the new hosts.

Use one of the following methods to add a new host:

[Using the Add Hosts Wizard to Add Hosts](#)

You can use the Add Hosts wizard to install CDH, Impala, and the Cloudera Manager Agent on a host.

[Disable TLS Encryption or Authentication](#)

If you have enabled TLS encryption or authentication for the Cloudera Manager Agents, you must disable both of them before starting the Add Hosts wizard. Otherwise, skip to the next step.



Important: This step leaves the existing hosts in an unmanageable state; they are still configured to use TLS, and so communicate with the Cloudera Manager Server.

1. From the **Administration** tab, select **Settings**.
2. Select the **Security** category.
3. Disable all levels of TLS that are currently enabled by deselecting the following options: **Use TLS Encryption for Agents**, and **Use TLS Authentication of Agents to Server**.
4. Click **Save Changes** to save the settings.
5. Restart the Cloudera Management Server to have these changes take effect.

Using the Add Hosts Wizard

1. Click the **Hosts** tab.
2. Click the **Add New Hosts** button.
3. Follow the instructions in the wizard to install the Oracle JDK and Cloudera Manager Agent packages and start the Agent.
4. In the **Specify hosts for your CDH Cluster installation** page, you can search for new hosts to add under the **New Hosts** tab. However, if you have hosts that are already known to Cloudera Manager but have no roles assigned, (for example, a host that was previously in your cluster but was then removed) these will appear under the **Currently Managed Hosts** tab.
5. You will have an opportunity to add (and start) role instances to your newly-added hosts using a host template.
 - a. You can select an existing host template, or create a new one.
 - b. To create a new host template, click the **+ Create...** button. This will open the **Create New Host Template** pop-up. See [Host Templates](#) on page 57 for details on how you select the role groups that define the roles that should run on a host. When you have created the template, it will appear in the list of host templates from which you can choose.
 - c. Select the host template you want to use.
 - d. By default Cloudera Manager will automatically start the roles specified in the host template on your newly added hosts. To prevent this, uncheck the option to start the newly-created roles.
6. When the wizard is finished, you can verify the Agent is connecting properly with the Cloudera Manager Server by clicking the **Hosts** tab and checking the health status for the new host. If the Health Status is **Good** and the value for the Last Heartbeat is recent, then the Agent is connecting properly with the Cloudera Manager Server.

If you did not specify a host template during the Add Hosts wizard, then no roles will be present on your new hosts until you add them. You can do this by adding individual roles under the **Instances** tab for a specific service, or by using a host template. See [Role Instances](#) on page 45 for information about adding roles for a specific service. See [Host Templates](#) on page 57 to create a host template that specifies a set of roles (from different services) that should run on a host.

Enable TLS Encryption or Authentication

If you previously enabled TLS security on your cluster, you must re-enable the TLS options on the **Administration** page and also configure TLS on each new host after using the Add Hosts wizard. Otherwise, you can ignore this step. For instructions, see [Configuring TLS Security for Cloudera Manager](#).

Enable SSL for CDH Components

If you have previously enabled SSL/TLS on your cluster, and you plan to start these roles on this new host, make sure you install the certificates that were being used by these roles on the source host. Since the new roles are inheriting their configuration from the previous host, make sure that the keystore or truststore passwords and locations are the same on the new host. For instructions on configuring SSL/TLS, see [Configuring SSL/TLS Encryption for CDH Services](#).

Enable Kerberos

If you have previously enabled Kerberos on your cluster:

- Install the packages required to `kinit` on the new host (see the list in [Before you Begin Using the Wizard](#)).
- If you have set up Cloudera Manager to manage `krb5.conf`, it will automatically deploy the file on the new host.
- If Cloudera Manager does not manage `krb5.conf`, you must manually update the file at `/etc/krb5.conf`.

Adding a Host by Installing the Packages Using Your Own Method

If you used a different mechanism to install the Oracle JDK, CDH, Cloudera Manager Agent packages, you can use that same mechanism to install the Oracle JDK, CDH, Cloudera Manager Agent packages and then start the Cloudera Manager Agent.

1. Install the Oracle JDK, CDH, and Cloudera Manager Agent packages using your own method. For instructions on installing these packages, see [Installation Path B - Manual Installation Using Cloudera Manager Packages](#).
2. After installation is complete, start the Cloudera Manager Agent. For instructions, see [Starting, Stopping, and Restarting Cloudera Manager Agents](#) on page 411.
3. After the Agent is started, you can verify the Agent is connecting properly with the Cloudera Manager Server by clicking the **Hosts** tab and checking the health status for the new host. If the Health Status is **Good** and the value for the Last Heartbeat is recent, then the Agent is connecting properly with the Cloudera Manager Server.
4. If you have enabled TLS security on your cluster, you must enable and configure TLS on each new host. Otherwise, ignore this step.
 - a. Enable and configure TLS on each new host by specifying `1` for the `use_tls` property in the `/etc/cloudera-scm-agent/config.ini` configuration file.
 - b. Configure the same level(s) of TLS security on the new hosts by following the instructions in [Configuring TLS Security for Cloudera Manager](#).
5. If you have previously enabled SSL/TLS on your cluster, and you plan to start these roles on this new host, make sure you install the certificates that were being used by these roles on the source host. Since the new roles are inheriting their configuration from the previous host, make sure that the keystore or truststore passwords and locations are the same on the new host. For instructions on configuring SSL/TLS, see [Configuring SSL/TLS Encryption for CDH Services](#).
6. If you have previously enabled Kerberos on your cluster:
 - Install the packages required to `kinit` on the new host (see the list in [Before you Begin Using the Wizard](#)).
 - If you have set up Cloudera Manager to manage `krb5.conf`, it will automatically deploy the file on the new host.
 - If Cloudera Manager does not manage `krb5.conf`, you must manually update the file at `/etc/krb5.conf`.

Specifying Racks for Hosts

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

To get maximum performance, it is important to configure CDH so that it knows the topology of your network. Network locations such as hosts and racks are represented in a tree, which reflects the network “distance” between locations. HDFS will use the network location to be able to place block replicas more intelligently to trade off performance and resilience. When placing jobs on hosts, CDH will prefer within-rack transfers (where there is more bandwidth available) to off-rack transfers; the MapReduce and YARN schedulers use network location to determine where the closest replica is as input to a map task. These computations are performed with the assistance of rack awareness scripts.

Cloudera Manager includes internal rack awareness scripts, but you must specify the racks where the hosts in your cluster are located. If your cluster contains more than 10 hosts, Cloudera recommends that you specify the rack for each host. HDFS, MapReduce, and YARN will automatically use the racks you specify.

Cloudera Manager supports nested rack specifications. For example, you could specify the rack `/rack3`, or `/group5/rack3` to indicate the third rack in the fifth group. All hosts in a cluster must have the *same number* of path components in their rack specifications.

To specify racks for hosts:

1. Click the **Hosts** tab.
2. Check the checkboxes next to the host(s) for a particular rack, such as all hosts for `/rack123`.
3. Click **Actions for Selected (n) > Assign Rack**, where *n* is the number of selected hosts.
4. Enter a rack name or ID that starts with a slash `/`, such as `/rack123` or `/aisle1/rack123`, and then click **Confirm**.
5. Optionally [restart affected services](#). Rack assignments are not automatically updated for running services.

Host Templates

Minimum Required Role: [Full Administrator](#)

Host templates let you designate a set of role groups that can be applied in a single operation to a host or a set of hosts. This significantly simplifies the process of configuring new hosts when you need to expand your cluster. Host templates are supported for both CDH 4 and CDH 5 cluster hosts.



Important: A host template can only be applied on a host with a version of CDH that matches the CDH version running on the cluster to which the host template belongs.

You can create and manage host templates under the Templates tab from the Hosts page.

1. Click the **Hosts** tab on the main Cloudera Manager navigation bar.
2. Click the **Templates** tab on the Hosts page.

Templates are not required; Cloudera Manager assigns roles and role groups to the hosts of your cluster when you perform the initial cluster installation. However, if you want to add new hosts to your cluster, a host template can make this much easier.

If there are existing host templates, they are listed on the page, along with links to each role group included in the template.

If you are managing multiple clusters, you must create separate host templates for each cluster, as the templates specify role configurations specific to the roles in a single cluster. Existing host templates are listed under the cluster to which they apply.

- You can click a role group name to be taken to the Edit configuration page for that role group, where you can modify the role group settings.
- From the **Actions** menu associated with the template you can edit the template, clone it, or delete it.

Creating a Host Template

1. From the **Templates** tab, click **Click here**
2. In the **Create New Host Template** pop-up window that appears:
 - Type a name for the template.
 - For each role, select the appropriate role group. There may be multiple role groups for a given role type — you want to select the one with the configuration that meets your needs.
3. Click **Create** to create the host template.

Editing a Host Template

1. From the **Hosts** tab, click the **Templates** tab.
2. Pull down the **Actions** menu for the template you want to modify, and click **Edit**. This put you into the **Edit Host Template** pop-up window. This works exactly like the **Create New Host Template** window — you can modify the template name or any of the role group selections.
3. Click **OK** when you have finished.

Applying a Host Template to a Host

You can use a host template to apply configurations for multiple roles in a single operation.

You can apply a template to a host that has no roles on it, or that has roles from the same services as those included in the host template. New roles specified in the template that do not already exist on the host will be added. A role on the host that is already a member of the role group specified in the template will be left unchanged. If a role on the host matches a role in the template, but is a member of a different role group, it will be moved to the role group specified by the template.

For example, suppose you have two role groups for a DataNode (DataNode Default Group and DataNode (1)). The host has a DataNode role that belongs to DataNode Default Group. If you apply a host template that specifies the DataNode (1) group, the role on the host will be moved from DataNode Default Group to DataNode (1).

However, if you have two instances of a service, such as MapReduce (for example, *mr1* and *mr2*) and the host has a TaskTracker role from service *mr2*, you cannot apply a TaskTracker role from service *mr1*.

A host may have no roles on it if you have just added the host to your cluster, or if you decommissioned a managed host and removed its existing roles.

Also, the host must have the same version of CDH installed as is running on the cluster whose host templates you are applying.

If a host belongs to a different cluster than the one for which you created the host template, you can apply the host template if the "foreign" host either has no roles on it, or has only management roles on it. When you apply the host template, the host will then become a member of the cluster whose host template you applied. The following instructions assume you have already created the appropriate host template.

1. Go to the **Hosts** page, **Status** tab.
2. Select the host(s) to which you want to apply your host template.
3. From the **Actions for Selected** menu, select **Apply Host Template**.
4. In the pop-up window that appears, select the host template you want to apply.
5. Optionally you can have Cloudera Manager start the roles created per the host template – check the box to enable this.
6. Click **Confirm** to initiate the action.

Maintenance Mode

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

Maintenance mode allows you to suppress alerts for a host, service, role, or an entire cluster. This can be useful when you need to take actions in your cluster (make configuration changes and restart various elements) and do not want to see the alerts that will be generated due to those actions.



Putting an entity into maintenance mode does not prevent events from being logged; it only suppresses the alerts that those events would otherwise generate. You can see a history of all the events that were recorded for entities during the period that those entities were in maintenance mode.

Explicit and Effective Maintenance Mode

When you enter maintenance mode on an entity (cluster, service, or host) that has subordinate entities (for example, the roles for a service) the subordinate entities are also put into maintenance mode. These are considered to be in **effective maintenance mode**, as they have inherited the setting from the higher-level entity.

For example:

- If you set the HBase service into maintenance mode, then its roles (HBase Master and all RegionServers) are put into effective maintenance mode.
- If you set a host into maintenance mode, then any roles running on that host are put into effective maintenance mode.

Entities that have been explicitly put into maintenance mode show the icon . Entities that have entered effective maintenance mode as a result of inheritance from a higher-level entity show the icon .

When an entity (role, host or service) is in effective maintenance mode, it can only be removed from maintenance mode when the higher-level entity exits maintenance mode. For example, if you put a service into maintenance mode, then the roles associated with that service will be entered into effective maintenance mode, and will remain in effective maintenance mode until the service exits maintenance mode. You cannot remove them from maintenance mode individually.

On the other hand, an entity that is in effective maintenance mode can be put into explicit maintenance mode. In this case, the entity will remain in maintenance mode even when the higher-level entity exits maintenance mode. For example, suppose you put a host into maintenance mode, (which puts all the roles on that host into effective maintenance mode). You then select one of the roles on that host and put it explicitly into maintenance mode. When you have the host exit maintenance mode, that one role will remain in maintenance mode. You will need to select it individually and specifically have it exit maintenance mode.

Viewing Maintenance Mode Status

You can view the status of Maintenance Mode in your cluster by clicking



to the right of the cluster name and selecting **View Maintenance Mode Status**.

Entering Maintenance Mode

You can enable maintenance mode for a cluster, service, role, or host.

Putting a Cluster into Maintenance Mode

1.

to the right of the cluster name and select **Enter Maintenance Mode**.

2. Confirm that you want to do this.

The cluster is put into explicit maintenance mode, as indicated by the icon. All services and roles in the cluster are entered into effective maintenance mode, as indicated by the icon.

Putting a Service into Maintenance Mode

1.

to the right of the service name and select **Enter Maintenance Mode**.

2. Confirm that you want to do this.

The service is put into explicit maintenance mode, as indicated by the icon. All roles for the service are entered into effective maintenance mode, as indicated by the icon.

Putting Roles into Maintenance Mode

1. Go to the service page that includes the role.
2. Go to the **Instances** tab.
3. Select the role(s) you want to put into maintenance mode.
4. From the **Actions for Selected** menu, select **Enter Maintenance Mode**.
5. Confirm that you want to do this.

The roles will be put in explicit maintenance mode. If the roles were already in effective maintenance mode (because its service or host was put into maintenance mode) the roles will now be in explicit maintenance mode. This means that they will not exit maintenance mode automatically if their host or service exits maintenance mode; they must be explicitly removed from maintenance mode.

Managing CDH and Managed Services

Putting a Host into Maintenance Mode


1. Go to the **Hosts** page.
2. Select the host(s) you want to put into maintenance mode.
3. From the **Actions for Selected** menu, select **Enter Maintenance Mode**.
4. Confirm that you want to do this.

The confirmation pop-up lists the role instances that will be put into effective maintenance mode when the host goes into maintenance mode.


Exiting Maintenance Mode

When you exit maintenance mode, the maintenance mode icons are removed and alert notification resumes.

Exiting a Cluster from Maintenance Mode

1.  to the right of the cluster name and select **Exit Maintenance Mode**.
2. Confirm that you want to do this.

Exiting a Service from Maintenance Mode

1.  to the right of the service name and select **Exit Maintenance Mode**.
2. Confirm that you want to do this.

Exiting Roles from Maintenance Mode

1. Go to the services page that includes the role.
2. Go to the **Instances** tab.
3. Select the role(s) you want to exit from maintenance mode.
4. From the **Actions for Selected** menu, select **Exit Maintenance Mode**.
5. Confirm that you want to do this.

Exiting a Host from Maintenance Mode

1. Go to the **Hosts** page.
2. Select the host(s) you want to put into maintenance mode.
3. From the **Actions for Selected** menu, select **Exit Maintenance Mode**.
4. Confirm that you want to do this.

The confirmation pop-up lists the role instances that will be removed from effective maintenance mode when the host exits maintenance mode.

Decommissioning and Recommissioning Hosts

Decommissioning a host decommissions and stops all roles on the host without having to go to each service and individually decommission the roles. Decommissioning applies to only to HDFS DataNode, MapReduce TaskTracker, YARN NodeManager, and HBase RegionServer roles. If the host has other roles running on it, those roles are stopped.

Once all roles on the host have been decommissioned and stopped, the host can be removed from service. You can decommission multiple hosts in parallel.

Decommissioning Hosts

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

You cannot decommission a DataNode or a host with a DataNode if the number of DataNodes equals the replication factor (which by default is three) of any file stored in HDFS. For example, if the replication factor of any file is three, and you have three DataNodes, you cannot decommission a DataNode or a host with a DataNode.



To decommission hosts:

1. If the host has a DataNode, perform the steps in [Tuning HDFS Prior to Decommissioning DataNodes](#) on page 61.
2. Click the **Hosts** tab.
3. Select the checkboxes next to one or more hosts.
4. Select **Actions for Selected > Decommission**.

A confirmation pop-up informs you of the roles that will be decommissioned or stopped on the hosts you have selected.

5. Click **Confirm**.

A Command Details window appears that will show each stop or decommission command as it is run, service by service. Click a decommission link to see the subcommands that are run for decommissioning a given role. Depending on the role, the steps may include adding the host to an "exclusions list" and refreshing the NameNode, JobTracker, or NodeManager; stopping the Balancer (if it is running); and moving data blocks or regions. Roles that do not have specific decommission actions are stopped.

While decommissioning is in progress, the host displays the  icon. After all roles have been decommissioned or stopped, the host displays the  icon. If one host in a cluster has been decommissioned, the DECOMMISSIONED facet displays in the Filters on the Hosts page, and you can filter the hosts according to their decommission status.

You cannot start roles on a decommissioned host.



Note: When a DataNode is decommissioned, the data blocks are not removed from the storage directories. You must delete the data manually.

Tuning HDFS Prior to Decommissioning DataNodes

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

When a DataNode is decommissioned, the NameNode ensures that every block from the DataNode will still be available across the cluster as dictated by the replication factor. This procedure involves copying blocks off the DataNode in small batches. In cases where a DataNode has thousands of blocks, decommissioning can take several hours. Before decommissioning hosts with DataNodes, you should first tune HDFS:

1. Raise the heap size of the DataNodes. DataNodes should be configured with at least 4 GB heap size to allow for the increase in iterations and max streams.
 - a. Go to the HDFS service page.
 - b. Click the **Configuration** tab.
 - c. Select **Scope > DataNode**.
 - d. Select **Category > Resource Management**.
 - e. Set the **Java Heap Size of DataNode in Bytes** property as recommended.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
 - f. Click **Save Changes** to commit the changes.
2. Set the DataNode balancing bandwidth:
 - a. Select **Scope > DataNode**.
 - b. Expand the **Category > Performance** category.
 - c. Configure the **DataNode Balancing Bandwidth** property to the bandwidth you have on your disks and network.
 - d. Click **Save Changes** to commit the changes.

3. Increase the replication work multiplier per iteration to a larger number (the default is 2, however 10 is recommended):
 - a. Select **Scope > NameNode**.
 - b. Expand the **Category > Advanced** category.
 - c. Configure the **Replication Work Multiplier Per Iteration** property to a value such as 10.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
 - d. Click **Save Changes** to commit the changes.
4. Increase the replication maximum threads and maximum replication thread hard limits:
 - a. Select **Scope > NameNode**.
 - b. Expand the **Category > Advanced** category.
 - c. Configure the **Maximum number of replication threads on a DataNode** and **Hard limit on the number of replication threads on a DataNode** properties to 50 and 100 respectively.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
 - d. Click **Save Changes** to commit the changes.
5. Restart the HDFS service.

Tuning HBase Prior to Decommissioning DataNodes

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


To increase the speed of a rolling restart of the HBase service, set the **Region Mover Threads** property to a higher value. This increases the number of regions that can be moved in parallel, but places additional strain on the HMaster. In most cases, **Region Mover Threads** should be set to 5 or lower.

Recommissioning Hosts

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

Only hosts that are decommissioned using Cloudera Manager can be recommissioned.

1. Click the **Hosts** tab.
2. Select one or more hosts to recommission.
3. Select **Actions for Selected > Recommission**.

The  icon is removed from the host and from the roles that reside on the host. However, the roles themselves are not restarted.

Restarting All The Roles on a Recommissioned Host

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

1. Click the **Hosts** tab.
2. Select one or more hosts on which to start recommissioned roles.
3. Select **Actions for Selected > Start All Roles**.

Deleting Hosts

Minimum Required Role: [Full Administrator](#)

You can remove a host from a cluster in two ways:

- Delete the host entirely from Cloudera Manager.
- Remove a host from a cluster, but leave it available to other clusters managed by Cloudera Manager.

Both methods [decommission the hosts](#), delete roles, and remove managed service software, but preserve data directories.

Deleting a Host from Cloudera Manager

1. In the Cloudera Manager Admin Console, click the **Hosts** tab.
2. Select the hosts to delete.
3. Select **Actions for Selected** > **Decommission**.
4. Stop the Agent on the host. For instructions, see [Starting, Stopping, and Restarting Cloudera Manager Agents](#) on page 411.
5. In the Cloudera Manager Admin Console, click the **Hosts** tab.
6. Reselect the hosts you selected in [step 2](#).
7. Select **Actions for Selected** > **Delete**.

Removing a Host From a Cluster

This procedure leaves the host managed by Cloudera Manager and preserves the Cloudera Management Service roles (such as the Events Server, Activity Monitor, and so on).

1. In the Cloudera Manager Admin Console, click the **Hosts** tab.
2. Select the hosts to delete.
3. Select **Actions for Selected** > **Remove From Cluster**. The Remove Hosts From Cluster dialog box displays.
4. Leave the selections to decommission roles and skip removing the Cloudera Management Service roles. Click **Confirm** to proceed with removing the selected hosts.

Managing Non-CDH Resources

From an operations perspective, CDH hosts may also run other processes, such as antivirus software or operating system backups. This topic presents information to help you plan for those processes.

Antivirus Software

If you use antivirus software on your servers, consider configuring it to skip scans on certain types of Hadoop-specific resources. It can take a long time to scan large files or directories with a large number of files. In addition, if your antivirus software locks files or directories as it scans them, those resources will be unavailable to your Hadoop processes during the scan, and can cause latency or unavailability of resources in your cluster. Consider skipping scans on the following types of resources:

- Scratch directories used by services such as Impala
- Log directories used by various Hadoop services
- Data directories which can grow to petabytes in size

The specific directory names and locations depend on the services your cluster uses and your configuration. In general, avoid scanning very large directories and filesystems. Instead, limit write access to these locations using security mechanisms such as access controls at the level of the operating system, HDFS, or at the service level.

Operating System Backups

Many of the considerations outlined in [Antivirus Software](#) on page 63 apply to operating system backups as well. Backing up scratch directories, log directories, and large amounts of data using standard operating system utilities may not make sense. In addition, many Hadoop resources cannot be backed up by conventional means due to their size and mutability. Consider excluding these types of resources from operating system backups, and using the techniques outlined in [Backup and Disaster Recovery](#) on page 377 instead.

If you use Cloudera Manager, it stores its configuration in a database. You should regularly perform backups of this database, using the mechanisms provided by the database vendor. See [Backing Up Databases](#).

Managing CDH from the Command Line

The following sections provide instructions and information on managing core Hadoop.

For installation and upgrade instructions, see the [#unique_130](#) guide, which also contains initial deployment and configuration instructions for core Hadoop and the CDH components, including:

- Cluster configuration and maintenance:
 - [Ports Used by Components of CDH 5](#)
 - [Configuring Network Names](#)
 - [Deploying CDH 5 on a Cluster](#)
 - [Starting CDH Services](#) on page 65
 - [Stopping CDH Services Using the Command Line](#) on page 70
- [Avro Usage](#)
- [Flume configuration](#)
- HBase configuration:
 - [Configuration Settings for HBase](#)
 - [HBase Replication](#) on page 380
 - [HBase Snapshots](#)
- [HCatalog configuration](#)
- [Impala configuration](#)
- Hive configuration:
 - [Configuring the Hive Metastore](#)
 - [Configuring HiveServer2](#)
 - [Configuring the Metastore to use HDFS High Availability](#)
- [HttpFS configuration](#)
- Hue: [Configuring CDH Components for Hue](#)
- Oozie configuration:
 - [Configuring Oozie](#)
- Parquet: [Using the Parquet File Format with Impala, Hive, Pig, and MapReduce](#)
- Snappy:
 - [Using Snappy for MapReduce Compression](#)
 - [Using Snappy for Pig Compression](#)
 - [Using Snappy for Hive Compression](#)
 - [Using Snappy Compression in Sqoop 1 and Sqoop 2 Imports](#)
 - [Using Snappy Compression with HBase](#)
- Spark configuration:
 - [Managing Spark Standalone Using the Command Line](#) on page 218
 - [Running Spark Applications](#) on page 224
 - [Running a Crunch Application with Spark](#) on page 232
- Sqoop configuration:
 - [Setting HADOOP_MAPRED_HOME](#) for Sqoop
 - [Configuring Sqoop 2](#)
- ZooKeeper: [Maintaining a ZooKeeper Server](#)

Starting CDH Services

You need to start and stop services in the right order to make sure everything starts or stops cleanly.



Note: The Oracle JDK is required for all Hadoop components.

START services in this order:

Order	Service	Comments	For instructions and more information
1	ZooKeeper	Cloudera recommends starting ZooKeeper before starting HDFS; this is a requirement in a high-availability (HA) deployment. In any case, always start ZooKeeper before HBase.	Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server ; Installing ZooKeeper in a Production Environment ; Deploying HDFS High Availability on page 299; Configuring High Availability for the JobTracker (MRv1)
2	HDFS	Start HDFS before all other services except ZooKeeper. If you are using HA, see the CDH 5 High Availability Guide for instructions.	Deploying HDFS on a Cluster ; HDFS High Availability on page 286
3	HttpFS		HttpFS Installation
4a	MRv1	Start MapReduce before Hive or Oozie. Do not start MRv1 if YARN is running.	Deploying MapReduce v1 (MRv1) on a Cluster ; Configuring High Availability for the JobTracker (MRv1)
4b	YARN	Start YARN before Hive or Oozie. Do not start YARN if MRv1 is running.	Deploying MapReduce v2 (YARN) on a Cluster
5	HBase		Starting and Stopping HBase on page 81; Deploying HBase in a Distributed Cluster
6	Hive	Start the Hive metastore before starting HiveServer2 and the Hive console.	Installing Hive
7	Oozie		Starting the Oozie Server
8	Flume 1.x		Running Flume
9	Sqoop		Sqoop Installation and Sqoop 2 Installation
10	Hue		Hue Installation

Configuring `init` to Start Hadoop System Services



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

`init(8)` starts some daemons when the system is booted. Depending on the distribution, `init` executes scripts from either the `/etc/init.d` directory or the `/etc/rc2.d` directory. The CDH packages link the files in `init.d` and `rc2.d` so that modifying one set of files automatically updates the other.

To start system services at boot time and on restarts, enable their `init` scripts on the systems on which the services will run, using the appropriate tool:

- `chkconfig` is included in the RHEL and CentOS distributions. Debian and Ubuntu users can install the `chkconfig` package.
- `update-rc.d` is included in the Debian and Ubuntu distributions.

Configuring `init` to Start Core Hadoop System Services in an MRv1 Cluster



Important:

Cloudera does not support running MRv1 and YARN daemons on the same nodes at the same time; it will degrade performance and may result in an unstable cluster deployment.

The `chkconfig` commands to use are:

```
$ sudo chkconfig hadoop-hdfs-namenode on
```

The `update-rc.d` commands to use on Ubuntu and Debian systems are:

Where	Command
On the NameNode	<pre>\$ sudo update-rc.d hadoop-hdfs-namenode defaults</pre>
On the JobTracker	<pre>\$ sudo update-rc.d hadoop-0.20-mapreduce-jobtracker defaults</pre>
On the Secondary NameNode (if used)	<pre>\$ sudo update-rc.d hadoop-hdfs-secondarynamenode defaults</pre>
On each TaskTracker	<pre>\$ sudo update-rc.d hadoop-0.20-mapreduce-tasktracker defaults</pre>
On each DataNode	<pre>\$ sudo update-rc.d hadoop-hdfs-datanode defaults</pre>

Configuring init to Start Core Hadoop System Services in a YARN Cluster



Important:

Do not run MRv1 and YARN on the same set of nodes at the same time. This is not recommended; it degrades your performance and may result in an unstable MapReduce cluster deployment.

The `chkconfig` commands to use are:

Where	Command
On the NameNode	<pre>\$ sudo chkconfig hadoop-hdfs-namenode on</pre>
On the ResourceManager	<pre>\$ sudo chkconfig hadoop-yarn-resourcemanager on</pre>
On the Secondary NameNode (if used)	<pre>\$ sudo chkconfig hadoop-hdfs-secondarynamenode on</pre>
On each NodeManager	<pre>\$ sudo chkconfig hadoop-yarn-nodemanager on</pre>
On each DataNode	<pre>\$ sudo chkconfig hadoop-hdfs-datanode on</pre>
On the MapReduce JobHistory node	<pre>\$ sudo chkconfig hadoop-mapreduce-historyserver on</pre>

The `update-rc.d` commands to use on Ubuntu and Debian systems are:

Where	Command
On the NameNode	<code>\$ sudo update-rc.d hadoop-hdfs-namenode defaults</code>
On the ResourceManager	<code>\$ sudo update-rc.d hadoop-yarn-resourcemanager defaults</code>
On the Secondary NameNode (if used)	<code>\$ sudo update-rc.d hadoop-hdfs-secondarynamenode defaults</code>
On each NodeManager	<code>\$ sudo update-rc.d hadoop-yarn-nodemanager defaults</code>
On each DataNode	<code>\$ sudo update-rc.d hadoop-hdfs-datanode defaults</code>
On the MapReduce JobHistory node	<code>\$ sudo update-rc.d hadoop-mapreduce-historyserver defaults</code>

Configuring init to Start Non-core Hadoop System Services

Non-core Hadoop daemons can also be configured to start at `init` time using the `chkconfig` or `update-rc.d` command.

The `chkconfig` commands are:

Component	Server	Command
Hue	Hue server	<code>\$ sudo chkconfig hue on</code>
Oozie	Oozie server	<code>\$ sudo chkconfig oozie on</code>
HBase	HBase master	<code>\$ sudo chkconfig hbase-master on</code>
	On each HBase RegionServer	<code>\$ sudo chkconfig hbase-regionserver on</code>
Hive Metastore	Hive Metastore server	<code>\$ sudo chkconfig hive-metastore on</code>
HiveServer2	HiveServer2	<code>\$ sudo chkconfig hive-server2 on</code>

Component	Server	Command
Zookeeper	Zookeeper server	<pre>\$ sudo chkconfig zookeeper-server on</pre>
HttpFS	HttpFS server	<pre>\$ sudo chkconfig hadoop-httpfs on</pre>

The `update-rc.d` commands to use on Ubuntu and Debian systems are:

Component	Server	Command
Hue	Hue server	<pre>\$ sudo update-rc.d hue defaults</pre>
Oozie	Oozie server	<pre>\$ sudo update-rc.d oozie defaults</pre>
HBase	HBase master	<pre>\$ sudo update-rc.d hbase-master defaults</pre>
	HBase RegionServer	<pre>\$ sudo update-rc.d hbase-regionserver defaults</pre>
Hive Metastore	Hive Metastore server	<pre>\$ sudo update-rc.d hive-metastore defaults</pre>
HiveServer2	HiveServer2	<pre>\$ sudo update-rc.d hive-server2 defaults</pre>
Zookeeper	Zookeeper server	<pre>\$ sudo update-rc.d zookeeper-server defaults</pre>
HttpFS	HttpFS server	<pre>\$ sudo update-rc.d hadoop-httpfs defaults</pre>

Starting and Stopping HBase Using the Command Line

When starting and stopping CDH services, order is important. See [Starting CDH Services](#) on page 65 and [Stopping CDH Services Using the Command Line](#) on page 70 for details. If you use Cloudera Manager, follow [these instructions](#) instead.



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

Starting HBase

When starting HBase, it is important to start the HMaster, followed by the RegionServers, then the Thrift server.

1. To start a HBase cluster using the command line, start the HBase Master by using the `sudo hbase-master start` command on RHEL or SuSE, or the `sudo hadoop-hbase-regionserver start` command on Ubuntu or Debian. The HMaster starts the RegionServers automatically.
2. To start a RegionServer manually, use the `sudo hbase-regionserver start` command on RHEL or SuSE, or the `sudo hadoop-hbase-regionserver start` command on Ubuntu or Debian.
3. To start the Thrift server, use the `hbase-thrift start` on RHEL or SuSE, or the `hadoop-hbase-thrift start` on Ubuntu or Debian.

Stopping HBase

When stopping HBase, it is important to stop the Thrift server, followed by each RegionServer, followed by any backup HMaster, and finally the main HMaster.

1. Shut down the Thrift server by using the `hbase-thrift stop` command on the Thrift server host. `sudo service hbase-thrift stop`
2. Shut down each RegionServer by using the `hadoop-hbase-regionserver stop` command on the RegionServer host.

```
sudo service hadoop-hbase-regionserver stop
```

3. Shut down backup HMaster, followed by the main HMaster, by using the `hbase-master stop` command.

```
sudo service hbase-master stop
```

Stopping CDH Services Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

To shut down all Hadoop Common system services (HDFS, YARN, MRv1), run the following on each host in the cluster:

```
$ for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done
```

To verify that no Hadoop processes are running, run the following command on each host in the cluster:

```
# ps -aef | grep java
```

To stop system services individually, use the instructions in the table below.



Important: Stop services in the order listed in the table. (You can start services in the reverse order.)


Order	Service	Comments	Instructions
1	Hue		<code>sudo service hue stop</code>
2	Impala		<code>sudo service impala-server stop</code> <code>sudo service impala-catalog stop</code> <code>sudo service impala-state-store stop</code>
3	Oozie		<code>sudo service oozie stop</code>
4	Hive	Exit the Hive console and ensure no Hive scripts are running. Stop the Hive server, HCatalog, and metastore daemon on each client.	<code>sudo service hiveserver2 stop</code> <code>sudo service hive-webhcat-server stop</code> <code>sudo service hive-metastore stop</code>
5	Flume 1.x	There is no Flume master.	<code>sudo service flume-ng-agent stop</code>
6	Sqoop 1		<code>sudo service sqoop-metastore stop</code>
6	Sqoop 2		<code>sudo service sqoop2-server stop</code>
7	Lily HBase Indexer (Solr/HBase Indexer)		<code>sudo service hbase-solr-indexer stop</code>
10	Spark		<code>sudo service spark-worker stop</code> <code>sudo service spark-history-server stop</code> <code>sudo service spark-master stop</code>
8	Sentry	Only present on a secure configuration.	<code>sudo service sentry-store stop</code>
9	Solr Search		<code>sudo service solr-server stop</code>
10	HBase	Stop the Thrift server and clients, followed by RegionServers and finally the Master.	<code>sudo service hbase-thrift stop</code> <code>sudo service hbase-rest stop</code> <code>sudo service hbase-regionserver stop</code> <code>sudo service hbase-master stop</code>
11	MapReduce v1	Stop the JobTracker service, then stop the TaskTracker	For MRv1 HA setup:

Order	Service	Comments	Instructions
		service on all nodes where it is running.	<pre>sudo service hadoop-0.20-mapreduce-jobtrackerha stop</pre> <p>For Non-HA setup:</p> <pre>sudo service hadoop-0.20-mapreduce-jobtracker stop</pre> <p>For all types of MRv1 setups:</p> <pre>sudo service hadoop-0.20-mapreduce-tasktracker stop</pre>
12	YARN	Stop the JobHistory server, followed by the ResourceManager and each of the NodeManagers.	<pre>\$ sudo service hadoop-mapreduce-historyserver stop</pre> <pre>\$ sudo service hadoop-yarn-resourcemanager stop</pre> <pre>\$ sudo service hadoop-yarn-nodemanager stop</pre>
13	HDFS	Stop HttpFS and the NFS Gateway (if present). Stop the Secondary NameNode, then the primary NameNode, followed by Journal nodes (if present) and then each of DataNodes.	<pre>sudo service hadoop-httpfs stop</pre> <pre>sudo service hadoop-hdfs-nfs3 stop</pre> <pre>sudo service hadoop-hdfs-secondarynamenode stop</pre> <pre>sudo service hadoop-hdfs-namenode stop</pre> <pre>sudo service hadoop-hdfs-journalnode stop</pre> <pre>sudo service hadoop-hdfs-datanode stop</pre>
14	KMS (Key Management Server)	Only present if HDFS at rest encryption is enabled	<pre>sudo service hadoop-kms-server stop</pre>
15	ZooKeeper		<pre>sudo service zookeeper-server stop</pre>

Migrating Data between Clusters Using distcp

You can migrate the data from any Hadoop cluster to a CDH 5 cluster by using a tool that copies out data in parallel, such as the DistCp tool offered in CDH 5. The following sections provide information and instructions:

Copying Data between two Clusters Using `distcp`

 **Important:** Do not run `distcp` as the HDFS user. The HDFS user is blacklisted for MapReduce jobs by default.

You can use the `distcp` tool on the destination cluster to initiate the copy job to move the data. Between two clusters running *different versions of CDH*, run the `distcp` tool with `hftp://` as the source file system and `hdfs://` as the destination file system. This uses the HFTP protocol for the source and the HDFS protocol for the destination. The default port for HFTP is 50070 and the default port for HDFS is 8020. Amazon S3 block and native filesystems are also supported, using the `s3a://` protocol.

Example of a source URI: `hftp://namenode-location:50070/basePath`

where `namenode-location` refers to the CDH 4 NameNode hostname as defined by its configured `fs.default.name` and 50070 is the NameNode's HTTP server port, as defined by the configured `dfs.http.address`.

Example of a destination URI: `hdfs://nameservice-id/basePath` or `hdfs://namenode-location`

This refers to the CDH 5 NameNode as defined by its configured `fs.defaultFS`.

The `basePath` in both the above URIs refers to the directory you want to copy, if one is specifically needed.

Example of an Amazon S3 Block Filesystem URI: `s3://accessKeyId:secretkey@bucket/file`

Example of an Amazon S3 Native Filesystem URI: `s3n://accessKeyId:secretkey@bucket/file`

The `distcp` Command

You can use `distcp` to copy files between compatible clusters in either direction, from or to the source or destination clusters.

For example, when upgrading, say from a CDH 5.7 cluster to a CDH 5.9, you should run `distcp` *from* the CDH 5.13 cluster in this manner:


```
$ hadoop distcp hftp://cdh57-namenode:50070/ hdfs://CDH59-nameservice/
$ hadoop distcp s3a://bucket/ hdfs://CDH59-nameservice/
```

You can also use a specific path, such as `/hbase` to move HBase data, for example:

```
$ hadoop distcp hftp://cdh57-namenode:50070/hbase hdfs://CDH59-nameservice/hbase
$ hadoop distcp s3a://bucket/file hdfs://CDH59-nameservice/bucket/file
```

Protocol Support for `distcp`

The following table lists support for using different protocols with the `distcp` command on different versions of CDH. In the table, **secure** means that the cluster is configured to use Kerberos. Copying between a secure cluster and an insecure cluster is only supported from CDH 5.1.3 onward, due to the inclusion of [HDFS-6776](#).

 **Note:** HFTP is a read-only protocol and cannot be used for the destination.

Source	Destination	Source protocol and configuration	Destination protocol and configuration	Where to issue <code>distcp</code> command	Fallback Configuration Required	Status
CDH 4	CDH 4	hdfs or webhdfs, insecure	hdfs or webhdfs, insecure	Source or destination		ok
CDH 4	CDH 4	hftp, insecure	hdfs or webhdfs, insecure	Destination		ok

Source	Destination	Source protocol and configuration	Destination protocol and configuration	Where to issue distcp command	Fallback Configuration Required	Status
CDH 4	CDH 4	hdfs or webhdfs, secure	hdfs or webhdfs, secure	Source or destination		ok
CDH 4	CDH 4	hftp, secure	hdfs or webhdfs, secure	Destination		ok
CDH 4	CDH 5	webhdfs, insecure	webhdfs or hdfs, insecure	Destination		ok
CDH 4	CDH 5 (5.1.3 and newer)	webhdfs, insecure	webhdfs, secure	Destination	yes	ok
CDH 4	CDH 5	webhdfs or hftp, insecure	webhdfs or hdfs, insecure	Destination		ok
CDH 4	CDH 5	webhdfs or hftp, secure	webhdfs or hdfs, secure	Destination		ok
CDH 4	CDH 5	hdfs or webhdfs, insecure	webhdfs, insecure	Source		ok
CDH 5	CDH 4	webhdfs , insecure	webhdfs, insecure	Source or destination		ok
CDH 5	CDH 4	webhdfs , insecure	hdfs, insecure	Destination		ok
CDH 5	CDH 4	hdfs, insecure	webhdfs, insecure	Source		ok
CDH 5	CDH 4	hftp, insecure	hdfs or webhdfs, insecure	Destination		ok
CDH 5	CDH 4	webhdfs, secure	webhdfs, secure	Source or destination		ok
CDH 5	CDH 4	webhdfs, secure	hdfs, insecure	Destination		ok
CDH 5	CDH 4	hdfs, secure	webhdfs, secure	Source		ok
CDH 5	CDH 5	hdfs or webhdfs, insecure	hdfs or webhdfs, insecure	Source or destination		ok
CDH 5	CDH 5	hftp, insecure	hdfs or webhdfs, insecure	Destination		ok
CDH 5	CDH 5	hdfs or webhdfs, secure	hdfs or webhdfs, secure	Source or destination		ok

Source	Destination	Source protocol and configuration	Destination protocol and configuration	Where to issue distcp command	Fallback Configuration Required	Status
CDH 5	CDH 5	hftp, secure	hdfs or webhdfs, secure	Destination		ok
CDH 5	CDH 5	hdfs or webhdfs, secure	webhdfs, insecure	Source	yes	ok
CDH 5	CDH 5	hdfs or webhdfs, insecure	hdfs or webhdfs, secure	Destination	yes	ok



Note: `distcp` does not support hftp from an insecure cluster to a secure cluster.

To enable the fallback configuration, for copying between a secure cluster and an insecure one, add the following to the HDFS `core-default.xml`, by using an advanced configuration snippet if you use Cloudera Manager, or editing the file directly otherwise.

```
<property>
  <name>ipc.client.fallback-to-simple-auth-allowed</name>
  <value>true</value>
</property>
```

`distcp` between Secure Clusters in Distinct Kerberos Realms

This section briefly describes the steps required to copy data between two secure clusters in distinct Kerberos realms.



Note: JDK version 1.7.x is required on both clusters when copying data between Kerberized clusters that are in different realms. For information about supported JDK versions, see [Supported JDK Versions](#).

Specify the Destination Parameters in `krb5.conf`

Edit the `krb5.conf` file on the client (where the `distcp` job will be submitted) to include the destination hostname and realm.

```
[realms]
HADOOP.QA.domain.COM = { kdc =kdc.domain.com:88 admin_server = admin.test.com:749
default_domain = domain.com supported_encetypes = arcfour-hmac:normal des-cbc-crc:normal
des-cbc-md5:normal des:normal des:v4 des:norealm des:onlyrealm des:afs3 }

[domain_realm]
.domain.com = HADOOP.test.domain.COM
domain.com = HADOOP.test.domain.COM
test03.domain.com = HADOOP.QA.domain.COM
```

(If SSL is enabled) Specify Truststore Properties

The following properties must be configured in the `ssl-client.xml` file on the client submitting the `distcp` job to establish trust between the target and destination clusters.

```
<property>
<name>ssl.client.truststore.location</name>
<value>path_to_truststore</value>
</property>

<property>
```

```
<name>ssl.client.truststore.password</name>
<value>XXXXXX</value>
</property>

<property>
<name>ssl.client.truststore.type</name>
<value>jks</value>
</property>
```

Set HADOOP_CONF to the Destination Cluster

Set the HADOOP_CONF path to be the destination environment. If you are not using HFTP, set the HADOOP_CONF path to the source environment instead.

Launch Distcp

Kinit on the client and launch the distcp job.

```
hadoop distcp hdfs://test01.domain.com:8020/user/alice
hdfs://test02.domain.com:8020/user/alice
```

If launching distcp fails, force Kerberos to use TCP instead of UDP by adding the following parameter to the krb5.conf file on the client.

```
[libdefaults]
udp_preference_limit = 1
```

Copying Data between a Secure and an Insecure Cluster using DistCp and WebHDFS

You can use DistCp and WebHDFS to copy data between a secure cluster and an insecure cluster by doing the following:

1. Set ipc.client.fallback-to-simple-auth-allowed to true in core-site.xml *on the secure cluster side*:

```
<property>
  <name>ipc.client.fallback-to-simple-auth-allowed</name>
  <value>>true</value>
</property>
```

2. Use commands such as the following *from the secure cluster side only*:

```
distcp webhdfs://insecureCluster webhdfs://secureCluster
distcp webhdfs://secureCluster webhdfs://insecureCluster
```

Post-migration Verification

After migrating data between the two clusters, it is a good idea to use `hadoop fs -ls /basePath` to verify the permissions, ownership and other aspects of your files, and correct any problems before using the files in your new cluster.

Managing Individual Services

The following sections cover the configuration and management of individual CDH and other services that have specific and unique requirements or options.

Managing Flume

The Flume packages are installed by the Installation wizard, but the service is not created. This page documents how to add, configure, and start the Flume service.

Adding a Flume Service

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

1. On the **Home > Status** tab, click



to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.

2. Select the **Flume** service and click **Continue**.
3. Select the radio button next to the services on which the new service should depend. All services must depend on the *same* ZooKeeper service. Click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

Configuring the Flume Agents

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

After you create a Flume service, you must first configure the agents before you start them. For detailed information about Flume agent configuration, see the [Flume User Guide](#).

The default Flume agent configuration provided in the **Configuration File** property of the **Agent** default role group is a configuration for a single agent in a single tier; you should replace this with your own configuration. When you add new agent roles, they are placed (initially) in the **Agent** default role group.

Agents that share the same configuration should be members of the same agent role group. You can create new [role groups](#) and can move agents between them. If your Flume configuration has multiple tiers, you must *create an agent role group for each tier*, and move each agent to be a member of the appropriate role group for their tier.

A Flume agent role group **Configuration File** property can contain the configuration for multiple agents, since each configuration property is prefixed by the agent name. You can [set the agents' names](#) using configuration overrides to change the name of a specific agent without changing its role group membership. Different agents can have the same name — agent names do not have to be unique.

1. Go to the Flume service.
2. Click the **Configuration** tab.
3. Select **Scope > Agent**. Settings you make to the default role group apply to all agent instances unless you associate those instances with a different role group, or override them for specific agents. See [Modifying Configuration Properties](#) on page 10.

4. Set the **Agent Name** property to the name of the agent (or one of the agents) defined in the `flume.conf` configuration file. The agent name can be comprised of letters, numbers, and the underscore character. You can specify only one agent name here — the name you specify will be used as the default for all Flume agent instances, unless you override the name for specific agents. You can have multiple agents with the same name — they will share the configuration specified in on the configuration file.
5. Copy the contents of the `flume.conf` file, in its entirety, into the **Configuration File** property. Unless overridden for specific agent instances, this property applies to all agents in the group. You can provide multiple agent configurations in this file and use agent name overrides to specify which configuration to use for each agent.



Important: The name-value property pairs in the **Configuration File** property *must* include an equal sign (=). For example, `tier1.channels.channel1.capacity = 10000`.

Setting a Flume Agent Name

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

If you have specified multiple agent configurations in a Flume agent role group **Configuration File** property, you can set the agent name for an agent that uses a different configuration. Overriding the agent name will point the agent to the appropriate properties specified in the agent configuration.

1. Go to the Flume service.
2. Click the **Configuration** tab.
3. Select **Scope > Agent**.
4. Locate the **Agent Name** property or search for it by typing its name in the Search box.
5. Enter a name for the agent.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Using Flume with HDFS or HBase Sinks

If you want to use Flume with HDFS or HBase sinks, you can add a dependency to that service from the Flume configuration page. This will automatically add the correct client configurations to the Flume agent's classpath.



Note: If you are using Flume with HBase, make sure that the `/etc/zookeeper/conf/zoo.cfg` file either does not exist on the host of the Flume agent that is using an HBase sink, or that it contains the correct ZooKeeper quorum.

Using Flume with Solr Sinks

Cloudera Manager provides a set of configuration settings under the Flume service to configure the Flume Morphline Solr Sink. See [Configuring the Flume Morphline Solr Sink for Use with the Solr Service](#) on page 216 for detailed instructions.

Updating Flume Agent Configurations

Minimum Required Role: [Full Administrator](#)

If you modify the **Configuration File** property after you have started the Flume service, update the configuration across Flume agents as follows:

1. Go to the Flume service.
2. Select **Actions > Update Config**.

Managing the HBase Service

Managing HBase

Cloudera Manager requires certain additional steps to set up and configure the HBase service.

Creating the HBase Root Directory

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

When adding the HBase service, the **Add Service** wizard automatically creates a root directory for HBase in HDFS. If you quit the **Add Service** wizard or it does not finish, you can create the root directory outside the wizard by doing these steps:

1. Choose **Create Root Directory** from the **Actions** menu in the **HBase > Status** tab.
2. Click **Create Root Directory** again to confirm.

Graceful Shutdown

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

A graceful shutdown of an HBase RegionServer allows the regions hosted by that RegionServer to be moved to other RegionServers before stopping the RegionServer. Cloudera Manager provides the following configuration options to perform a graceful shutdown of either an HBase RegionServer or the entire service.

To increase the speed of a rolling restart of the HBase service, set the **Region Mover Threads** property to a higher value. This increases the number of regions that can be moved in parallel, but places additional strain on the HMaster. In most cases, **Region Mover Threads** should be set to 5 or lower.

Gracefully Shutting Down an HBase RegionServer

1. Go to the HBase service.
2. Click the **Instances** tab.
3. From the list of Role Instances, select the RegionServer you want to shut down gracefully.
4. Select **Actions for Selected > Decommission (Graceful Stop)**.
5. Cloudera Manager attempts to gracefully shut down the RegionServer for the interval configured in the [Graceful Shutdown Timeout](#) configuration option, which defaults to 3 minutes. If the graceful shutdown fails, Cloudera Manager forcibly stops the process by sending a `SIGKILL (kill -9)` signal. HBase will perform recovery actions on regions that were on the forcibly stopped RegionServer.
6. If you cancel the graceful shutdown before the **Graceful Shutdown Timeout** expires, you can still manually stop a RegionServer by selecting **Actions for Selected > Stop**, which sends a `SIGTERM (kill -5)` signal.

Gracefully Shutting Down the HBase Service

1. Go to the HBase service.
2. Select **Actions > Stop**. This tries to perform an HBase Master-driven graceful shutdown for the length of the configured Graceful Shutdown Timeout (three minutes by default), after which it abruptly shuts down the whole service.

Configuring the Graceful Shutdown Timeout Property

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

This timeout only affects a graceful shutdown of the entire HBase service, not individual RegionServers. Therefore, if you have a large cluster with many RegionServers, you should strongly consider increasing the timeout from its default of 180 seconds.

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > HBASE-1 (Service Wide)**
4. Use the Search box to search for the **Graceful Shutdown Timeout** property and edit the value.

5. Click **Save Changes** to save this setting.

Configuring the HBase Thrift Server Role

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

The Thrift Server role is not added by default when you install HBase, but it is required before you can use certain other features such as the Hue HBase browser. To add the Thrift Server role:

1. Go to the HBase service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Select the host(s) where you want to add the Thrift Server role (you only need one for Hue) and click **Continue**.
The Thrift Server role should appear in the instances list for the HBase server.
5. Select the Thrift Server role instance.
6. Select **Actions for Selected > Start**.

Enabling HBase Indexing

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

HBase indexing is dependent on the [Key-Value Store Indexer service](#). The Key-Value Store Indexer service uses the [Lily HBase Indexer Service](#) to index the stream of records being added to HBase tables. Indexing allows you to query data stored in HBase with the [Solr service](#).

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > HBASE-1 (Service Wide)**
4. Select **Category > Backup**.
5. Select the **Enable Replication** and **Enable Indexing** properties.
6. Click **Save Changes**.

Adding a Custom Coprocessor

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



Note: End-user custom HBase coprocessors are not supported.

The HBase coprocessor framework provides a way to extend HBase with custom functionality. To configure these properties in Cloudera Manager:

1. Select the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > All**.
4. Select **Category > All**.
5. Type `HBase Coprocessor` in the Search box.
6. You can configure the values of the following properties:
 - **HBase Coprocessor Abort on Error** (Service-Wide)
 - **HBase Coprocessor Master Classes** (Master Default Group)
 - **HBase Coprocessor Region Classes** (RegionServer Default Group)
7. Click **Save Changes** to commit the changes.

Enabling Hedged Reads on HBase

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

1. Go to the HBase service.
2. Click the **Configuration** tab.

3. Select **Scope > HBASE-1 (Service-Wide)**.
4. Select **Category > Performance**.
5. Configure the **HDFS Hedged Read Threadpool Size** and **HDFS Hedged Read Delay Threshold** properties. The descriptions for each of these properties on the configuration pages provide more information.
6. Click **Save Changes** to commit the changes.

Advanced Configuration for Write-Heavy Workloads

HBase includes several advanced configuration parameters for adjusting the number of threads available to service flushes and compactions in the presence of write-heavy workloads. Tuning these parameters incorrectly can severely degrade performance and is not necessary for most HBase clusters. If you use Cloudera Manager, configure these options using the **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml**.

`hbase.hstore.flusher.count`

The number of threads available to flush writes from memory to disk. Never increase `hbase.hstore.flusher.count` to more of 50% of the number of disks available to HBase. For example, if you have 8 solid-state drives (SSDs), `hbase.hstore.flusher.count` should never exceed 4. This allows scanners and compactions to proceed even in the presence of very high writes.

`hbase.regionserver.thread.compaction.large` and `hbase.regionserver.thread.compaction.small`

The number of threads available to handle small and large compactions, respectively. Never increase either of these options to more than 50% of the number of disks available to HBase.

Ideally, `hbase.regionserver.thread.compaction.small` should be greater than or equal to `hbase.regionserver.thread.compaction.large`, since the large compaction threads do more intense work and will be in use longer for a given operation.

In addition to the above, if you use compression on some column families, more CPU will be used when flushing these column families to disk during flushes or compaction. The impact on CPU usage depends on the size of the flush or the amount of data to be decompressed and compressed during compactions.

Starting and Stopping HBase

Use these instructions to start, stop, restart, rolling restart, or decommission HBase clusters or individual hosts.

Starting or Restarting HBase

You can start HBase hosts individually or as an entire cluster.

Starting or Restarting HBase Using Cloudera Manager

1. Go to the HBase service.
2. Click the **Actions** button and select **Start**.
3. To restart a running cluster, click **Actions** and select **Restart** or **Rolling Restart**. A rolling restart, which restarts each RegionServer, one at a time, after a grace period. To configure the grace period, see [Configuring the Graceful Shutdown Timeout Property](#) on page 79.
4. The Thrift service has no dependencies and can be restarted at any time. To stop or restart the Thrift service:
 - Go to the HBase service.
 - Select Instances.
 - Select the **HBase Thrift Server** instance.
 - Select **Actions for Selected** and select either **Stop** or **Restart**.

Starting or Restarting HBase Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

If you need the ability to perform a rolling restart, Cloudera recommends managing your cluster with Cloudera Manager.

1. To start a HBase cluster using the command line, start the HBase Master by using the `sudo hbase-master start` command on RHEL or SuSE, or the `sudo hadoop-hbase-regionserver start` command on Ubuntu or Debian. The HMaster starts the RegionServers automatically.
2. To start a RegionServer manually, use the `sudo hbase-regionserver start` command on RHEL or SuSE, or the `sudo hadoop-hbase-regionserver start` command on Ubuntu or Debian. Running multiple RegionServer processes on the same host is not supported.
3. The Thrift service has no dependencies and can be restarted at any time. To start the Thrift server, use the `hbase-thrift start` on RHEL or SuSE, or the `hadoop-hbase-thrift start` on Ubuntu or Debian.

Stopping HBase

You can stop a single HBase host, all hosts of a given type, or all hosts in the cluster.

Stopping HBase Using Cloudera Manager

1. To stop or decommission a single RegionServer:
 - a. Go to the HBase service.
 - b. Click the **Instances** tab.
 - c. From the list of Role Instances, select the RegionServer or RegionServers you want to stop or decommission.
 - d. Select **Actions for Selected** and select either **Decommission (Graceful Stop)** or **Stop**.
 - **Graceful Stop** causes the regions to be redistributed to other RegionServers, increasing availability during the RegionServer outage. Cloudera Manager waits for an interval determined by the [Graceful Shutdown timeout](#) interval, which defaults to three minutes. If the graceful stop does not succeed within this interval, the RegionServer is stopped with a `SIGKILL (kill -9)` signal. Recovery will be initiated on affected regions.
 - **Stop** happens immediately and does not redistribute the regions. It issues a `SIGTERM (kill -5)` signal.
2. To stop or decommission a single HMaster, select the Master and go through the same steps as above.
3. To stop or decommission the entire cluster, select the **Actions** button at the top of the screen (not **Actions for selected**) and select **Decommission (Graceful Stop)** or **Stop**.

Stopping HBase Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Shut down the Thrift server by using the `hbase-thrift stop` command on the Thrift server host. `sudo service hbase-thrift stop`

2. Shut down each RegionServer by using the `hadoop-hbase-regionserver stop` command on the RegionServer host.

```
sudo service hadoop-hbase-regionserver stop
```

3. Shut down backup HMaster, followed by the main HMaster, by using the `hbase-master stop` command.

```
sudo service hbase-master stop
```

Configuring the HBase Canary

The HBase canary is an optional service that periodically checks that a RegionServer is alive. This canary is different from the Cloudera Service Monitoring canary and is provided by the HBase service. The HBase canary is disabled by default. After enabling the canary, you can configure several different thresholds and intervals relating to it, as well as exclude certain tables from the canary checks. The canary works on Kerberos-enabled clusters if you have the HBase client configured to use Kerberos.

Configure the HBase Canary Using Cloudera Manager

Minimum Required Role: [Full Administrator](#)

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope** > **HBase or HBase Service-Wide**.
4. Select **Category** > **Monitoring**.
5. Locate the **HBase Canary** property or search for it by typing its name in the Search box. Several properties have *Canary* in the property name.
6. Select the checkbox.
7. Review other HBase Canary properties to configure the specific behavior of the canary.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

8. Click **Save Changes** to commit the changes.
9. Restart the role.
10. Restart the service.

Configure the HBase Canary Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

The HBase canary is a Java class. To run it from the command line, in the foreground, issue a command similar to the following, as the HBase user:

```
$ /usr/bin/hbase org.apache.hadoop.hbase.tool.Canary
```

To start the canary in the background, add the `--daemon` option. You can also use this option in your HBase startup scripts.

```
$ /usr/bin/hbase org.apache.hadoop.hbase.tool.Canary --daemon
```

The canary has many options. To see usage instructions, add the `--help` parameter:

```
$ /usr/bin/hbase org.apache.hadoop.hbase.tool.Canary --help
```

Checking and Repairing HBase Tables

HBaseFsk (`hbck`) is a command-line tool that checks for region consistency and table integrity problems and repairs corruption. It works in two basic modes — a read-only inconsistency identifying mode and a multi-phase read-write repair mode.

- **Read-only inconsistency identification:** In this mode, which is the default, a report is generated but no repairs are attempted.
- **Read-write repair mode:** In this mode, if errors are found, `hbck` attempts to repair them.

You can run `hbck` manually or configure the `hbck` poller to run `hbck` periodically.

Always run HBase administrative commands such as the HBase Shell, `hbck`, or bulk-load commands as the HBase user (typically `hbase`).

Running `hbck` Manually

The `hbck` command is located in the `bin` directory of the HBase install.

- With no arguments, `hbck` checks HBase for inconsistencies and prints OK if no inconsistencies are found, or the number of inconsistencies otherwise.
- With the `-details` argument, `hbck` checks HBase for inconsistencies and prints a detailed report.
- To limit `hbck` to only checking specific tables, provide them as a space-separated list: `hbck <table1> <table2>`



Warning: The following `hbck` options modify HBase metadata and are dangerous. They are not coordinated by the HMaster and can cause further corruption by conflicting with commands that are currently in progress or coordinated by the HMaster. Even if the HMaster is down, it may try to recover the latest operation when it restarts. These options should only be used as a last resort. The `hbck` command can only fix actual HBase metadata corruption and is not a general-purpose maintenance tool. Before running these commands, consider contacting Cloudera Support for guidance. In addition, running any of these commands requires a HMaster restart.

- If region-level inconsistencies are found, use the `-fix` argument to direct `hbck` to try to fix them. The following sequence of steps is followed:
 1. The standard check for inconsistencies is run.
 2. If needed, repairs are made to tables.
 3. If needed, repairs are made to regions. Regions are closed during repair.
- You can also fix individual region-level inconsistencies separately, rather than fixing them automatically with the `-fix` argument.
 - `-fixAssignments` repairs unassigned, incorrectly assigned or multiply assigned regions.
 - `-fixMeta` removes rows from `hbase:meta` when their corresponding regions are not present in HDFS and adds new meta rows if regions are present in HDFS but not in `hbase:meta`.
 - `-repairHoles` creates HFiles for new empty regions on the filesystem and ensures that the new regions are consistent.
 - `-fixHdfsOrphans` repairs a region directory that is missing a region metadata file (the `.regioninfo` file).
 - `-fixHdfsOverlaps` fixes overlapping regions. You can further tune this argument using the following options:
 - `-maxMerge <n>` controls the maximum number of regions to merge.
 - `-sidelineBigOverlaps` attempts to sideline the regions which overlap the largest number of other regions.
 - `-maxOverlapsToSideline <n>` limits the maximum number of regions to sideline.
- To try to repair all inconsistencies and corruption at once, use the `-repair` option, which includes all the region and table consistency options.

For more details about the `hbck` command, see [Appendix C](#) of the HBase Reference Guide.

Configuring the hbck Poller

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

The hbck poller is a feature of Cloudera Manager, which can be configured to run hbck automatically, in read-only mode, and send alerts if errors are found. By default, it runs every 30 minutes. Several configuration settings are available for the hbck poller. The hbck poller is not provided if you use CDH without Cloudera Manager.

Configuring the hbck Poller

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > All or HBase Service-Wide**.
4. Select **Category > Monitoring**.
5. Verify that the **HBase Hbck Poller** checkbox is checked.
6. Locate each property or search for it by typing its name in the Search box.
7. Configure the alert behavior with the following settings:
 - **HBase Hbck Poller Maximum Error Count:** The maximum number of errors that the hbck poller will retain through a given run.
 - **HBase Hbck Region Error Count:** An alert is published if at least this number of regions is detected with errors across all regions in this service. If the value is not set, alerts will not be published based on the count of regions with errors.
 - **Alert Threshold:** An alert is published if the number of errors reaches this threshold.
 - **HBase Hbck Error Count Alert Threshold:** An alert is published if at least this number of tables is detected with errors across all tables in this service. Some errors are not associated with a region, such as `RS_CONNECT_FAILURE`. If the value is not set, alerts will not be published based on the count of tables with errors.
 - **HBase Hbck Alert Error Codes:** An alert is published errors match any of the specified codes. The default behavior is not to limit the error codes which trigger an alert. May be set to one or more of the following:
 - UNKNOWN
 - NO_META_REGION
 - NULL_ROOT_REGION
 - NO_VERSION_FILE
 - NOT_IN_META_HDFS
 - NOT_IN_META
 - NOT_IN_META_OR_DEPLOYED
 - NOT_IN_HDFS_OR_DEPLOYED
 - NOT_IN_HDFS
 - SERVER_DOES_NOT_MATCH_META
 - NOT_DEPLOYED
 - MULTI_DEPLOYED
 - SHOULD_NOT_BE_DEPLOYED
 - MULTI_META_REGION
 - RS_CONNECT_FAILURE
 - FIRST_REGION_STARTKEY_NOT_EMPTY
 - LAST_REGION_ENDKEY_NOT_EMPTY
 - DUPE_STARTKEYS
 - HOLE_IN_REGION_CHAIN
 - OVERLAP_IN_REGION_CHAIN
 - REGION_CYCLE
 - DEGENERATE_REGION
 - ORPHAN_HDFS_REGION

- LINGERING_SPLIT_PARENT
- NO_TABLEINFO_FILE

8. To configure the polling interval, edit the **Service Monitor Derived Configs Advanced Configuration Snippet** with a setting such as the following, which sets the polling interval to 60 minutes. Restart the RegionServers for the changes to take effect.

```
<property>
  <name>smon.hbase.fsckpoller.interval.ms</name>
  <value>3600000</value>
</property>
```

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

Hedged Reads

Hadoop 2.4 introduced a new feature called *hedged reads*. If a read from a block is slow, the HDFS client starts up another parallel, 'hedged' read against a different block replica. The result of whichever read returns first is used, and the outstanding read is cancelled. This feature helps in situations where a read occasionally takes a long time rather than when there is a systemic problem. Hedged reads can be enabled for HBase when the HFiles are stored in HDFS. This feature is disabled by default.

Enabling Hedged Reads for HBase Using Cloudera Manager

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

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > HBASE-1 (Service-Wide)**.
4. Select **Category > Performance**.
5. Configure the **HDFS Hedged Read Threadpool Size** and **HDFS Hedged Read Delay Threshold** properties. The descriptions for each of these properties on the configuration pages provide more information.
6. Click **Save Changes** to commit the changes.

Enabling Hedged Reads for HBase Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

To enable hedged reads for HBase, edit the `hbase-site.xml` file on each server. Set `dfs.client.hedged.read.threadpool.size` to the number of threads to dedicate to running hedged threads, and set the `dfs.client.hedged.read.threshold.millis` configuration property to the number of milliseconds to wait before starting a second read against a different block replica. Set `dfs.client.hedged.read.threadpool.size` to 0 or remove it from the configuration to disable the feature. After changing these properties, restart your cluster.

The following is an example configuration for hedged reads for HBase.

```
<property>
  <name>dfs.client.hedged.read.threadpool.size</name>
  <value>20</value> <!-- 20 threads -->
</property>
<property>
  <name>dfs.client.hedged.read.threshold.millis</name>
  <value>10</value> <!-- 10 milliseconds -->
</property>
```

Monitoring the Performance of Hedged Reads

You can monitor the performance of hedged reads using the following metrics emitted by Hadoop when hedged reads are enabled.

- **hedgedReadOps** - the number of hedged reads that have occurred
- **hedgeReadOpsWin** - the number of times the hedged read returned faster than the original read

Configuring the Blocksize for HBase

The blocksize is an important configuration option for HBase. HBase data is stored in one (after a major compaction) or more (possibly before a major compaction) HFiles per column family per region. It determines both of the following:

- The blocksize for a given column family determines the smallest unit of data HBase can read from the column family's HFiles.
- It is also the basic unit of measure cached by a RegionServer in the BlockCache.

The default blocksize is 64 KB. The appropriate blocksize is dependent upon your data and usage patterns. Use the following guidelines to tune the blocksize size, in combination with testing and benchmarking as appropriate.



Warning: The default blocksize is appropriate for a wide range of data usage patterns, and tuning the blocksize is an advanced operation. The wrong configuration can negatively impact performance.

- Consider the average key/value size for the column family when tuning the blocksize. You can find the average key/value size using the HFile utility:

```
$ hbase org.apache.hadoop.hbase.io.hfile.HFile -f /path/to/HFILE -m -v
...
Block index size as per heapsize: 296
reader=hdfs://srv1.example.com:9000/path/to/HFILE, \
compression=none, inMemory=false, \
firstKey=US6683275_20040127/mimetype:/1251853756871/Put, \
lastKey=US6684814_20040203/mimetype:/1251864683374/Put, \
avgKeyLen=37, avgValueLen=8, \
entries=1554, length=84447
...
```

- Consider the pattern of reads to the table or column family. For instance, if it is common to scan for 500 rows on various parts of the table, performance might be increased if the blocksize is large enough to encompass 500-1000 rows, so that often, only one read operation on the HFile is required. If your typical scan size is only 3 rows, returning 500-1000 rows would be overkill.

It is difficult to predict the size of a row before it is written, because the data will be compressed when it is written to the HFile. Perform testing to determine the correct blocksize for your data.

Configuring the Blocksize for a Column Family

You can configure the blocksize of a column family at table creation or by disabling and altering an existing table. These instructions are valid whether or not you use Cloudera Manager to manage your cluster.

```
hbase> create 'test_table', {NAME => 'test_cf', BLOCKSIZE => '262144'}
hbase> disable 'test_table'
hbase> alter 'test_table', {NAME => 'test_cf', BLOCKSIZE => '524288'}
hbase> enable 'test_table'
```

After changing the blocksize, the HFiles will be rewritten during the next major compaction. To trigger a major compaction, issue the following command in HBase Shell.

```
hbase> major_compact 'test_table'
```

Depending on the size of the table, the major compaction can take some time and have a performance impact while it is running.

Monitoring Blocksize Metrics

Several metrics are exposed for monitoring the blocksize by monitoring the blockcache itself.

Configuring the HBase BlockCache

In the default configuration, HBase uses a single on-heap cache. If you configure the off-heap `BucketCache`, the on-heap cache is used for Bloom filters and indexes, and the off-heap `BucketCache` is used to cache data blocks. This is referred to as the **Combined** Blockcache configuration. The Combined `BlockCache` allows you to use a larger in-memory cache while reducing the negative impact of garbage collection in the heap, because HBase manages the `BucketCache`, rather than relying on the garbage collector.

Contents of the BlockCache

In order to size the `BlockCache` correctly, you need to understand what HBase places into it.

- **Your data:** Each time a Get or Scan operation occurs, the result is added to the `BlockCache` if it was not already cached there. If you use the `BucketCache`, data blocks are always cached in the `BucketCache`.
- **Row keys:** When a value is loaded into the cache, its row key is also cached. This is one reason to make your row keys as small as possible. A larger row key takes up more space in the cache.
- **hbase:meta:** The `hbase:meta` catalog table keeps track of which `RegionServer` is serving which regions. It can consume several megabytes of cache if you have a large number of regions, and has `in-memory` access priority, which means HBase attempts to keep it in the cache as long as possible.
- **Indexes of HFiles:** HBase stores its data in HDFS in a format called *HFile*. These HFiles contain indexes which allow HBase to seek for data within them without needing to open the entire HFile. The size of an index is a factor of the block size, the size of your row keys, and the amount of data you are storing. For big data sets, the size can exceed 1 GB per `RegionServer`, although the entire index is unlikely to be in the cache at the same time. If you use the `BucketCache`, indexes are always cached on-heap.
- **Bloom filters:** If you use Bloom filters, they are stored in the `BlockCache`. If you use the `BucketCache`, Bloom filters are always cached on-heap.

The sum of the sizes of these objects is highly dependent on your usage patterns and the characteristics of your data. For this reason, the HBase Web UI and Cloudera Manager each expose several metrics to help you size and tune the `BlockCache`.

Deciding Whether To Use the BucketCache

The HBase team has published the [results of exhaustive BlockCache testing](#), which revealed the following guidelines.

- If the result of a Get or Scan typically fits completely in the heap, the default configuration, which uses the on-heap `LruBlockCache`, is the best choice, as the L2 cache will not provide much benefit. If the eviction rate is low, garbage collection can be 50% less than that of the `BucketCache`, and throughput can be at least 20% higher.
- Otherwise, if your cache is experiencing a consistently high eviction rate, use the `BucketCache`, which causes 30-50% of the garbage collection of `LruBlockCache` when the eviction rate is high.
- `BucketCache` using *file mode* on solid-state disks has a better garbage-collection profile but lower throughput than `BucketCache` using *off-heap memory*.

Bypassing the BlockCache

If the data needed for a specific but atypical operation does not all fit in memory, using the `BlockCache` can be counter-productive, because data that you are still using may be evicted, or even if other data is not evicted, excess garbage collection can adversely effect performance. For this type of operation, you may decide to bypass the `BlockCache`. To bypass the `BlockCache` for a given Scan or Get, use the `setCacheBlocks(false)` method.

In addition, you can prevent a specific column family's contents from being cached, by setting its `BLOCKCACHE` configuration to `false`. Use the following syntax in HBase Shell:

```
hbase> alter 'myTable', CONFIGURATION => {NAME => 'myCF', BLOCKCACHE => 'false'}
```


Cache Eviction Priorities

Both the on-heap cache and the off-heap `BucketCache` use the same cache priority mechanism to decide which cache objects to evict in order to make room for new objects. Three levels of block priority allow for scan-resistance and in-memory column families. Objects evicted from the cache are subject to garbage collection.

- **Single access priority:** The first time a block is loaded from HDFS, that block is given single access priority, which means that it will be part of the first group to be considered during evictions. Scanned blocks are more likely to be evicted than blocks that are used more frequently.
- **Multi access priority:** If a block in the single access priority group is accessed again, that block is assigned multi access priority, which moves it to the second group considered during evictions, and is therefore less likely to be evicted.
- **In-memory access priority:** If the block belongs to a column family which is configured with the `in-memory` configuration option, its priority is changed to in memory access priority, regardless of its access pattern. This group is the last group considered during evictions, but is not guaranteed not to be evicted. Catalog tables are configured with in-memory access priority.

To configure a column family for in-memory access, use the following syntax in HBase Shell:

```
hbase> alter 'myTable', 'myCF', CONFIGURATION => {IN_MEMORY => 'true'}
```

To use the Java API to configure a column family for in-memory access, use the `HColumnDescriptor.setInMemory(true)` method.

Sizing the BlockCache

When you use the `LruBlockCache`, the blocks needed to satisfy each read are cached, evicting older cached objects if the `LruBlockCache` is full. The size cached objects for a given read may be significantly larger than the actual result of the read. For instance, if HBase needs to scan through 20 HFile blocks to return a 100 byte result, and the HFile blocksize is 100 KB, the read will add $20 * 100$ KB to the `LruBlockCache`.

Because the `LruBlockCache` resides entirely within the Java heap, the amount of which is available to HBase and what percentage of the heap is available to the `LruBlockCache` strongly impact performance. By default, the amount of HBase heap reserved for `LruBlockCache` (`hfile.block.cache.size`) is `.40`, or 40%. To determine the amount of heap available for the `LruBlockCache`, use the following formula. The `0.99` factor allows 1% of heap to be available as a "working area" for evicting items from the cache. If you use the `BucketCache`, the on-heap `LruBlockCache` only stores indexes and Bloom filters, and data blocks are cached in the off-heap `BucketCache`.

```
number of RegionServers * heap size * hfile.block.cache.size * 0.99
```

To tune the size of the `LruBlockCache`, you can add `RegionServers` or increase the total Java heap on a given `RegionServer` to increase it, or you can tune `hfile.block.cache.size` to reduce it. Reducing it will cause cache evictions to happen more often, but will reduce the time it takes to perform a cycle of garbage collection. Increasing the heap will cause garbage collection to take longer but happen less frequently.

About the off-heap `BucketCache`

If the `BucketCache` is enabled, it stores data blocks, leaving the on-heap cache free for storing indexes and Bloom filters. The physical location of the `BucketCache` storage can be either in memory (off-heap) or in a file stored in a fast disk.

- **Off-heap:** This is the default configuration.
- **File-based:** You can use the file-based storage mode to store the `BucketCache` on an SSD or FusionIO device,

Starting in CDH 5.4 (HBase 1.0), you can configure a column family to keep its data blocks in the L1 cache instead of the `BucketCache`, using the `HColumnDescriptor.cacheDataInL1(true)` method or by using the following syntax in HBase Shell:

```
hbase> alter 'myTable', CONFIGURATION => {CACHE_DATA_IN_L1 => 'true'}}
```

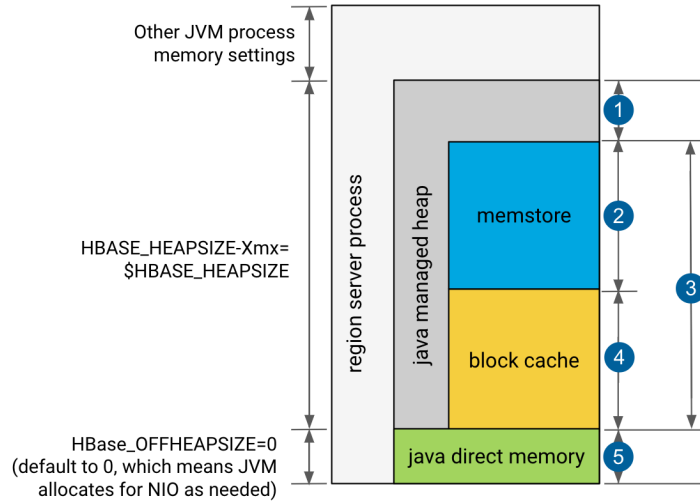
Configuring the off-heap BucketCache

This table summarizes the important configuration properties for the BucketCache. To configure the BucketCache, see [Configuring the Off-heap BucketCache Using Cloudera Manager](#) on page 92 or [Configuring the Off-heap BucketCache Using the Command Line](#) on page 93. The table is followed by three diagrams that show the impacts of different blockcache settings.

Table 1: BucketCache Configuration Properties

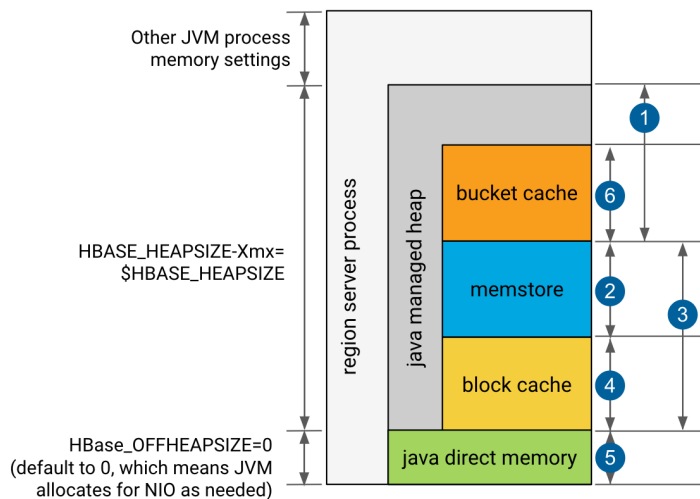
Property	Default	Description
<code>hbase.bucketcache.combinedcache.enabled</code>	true	When BucketCache is enabled, use it as a L2 cache for LruBlockCache. If set to true, indexes and Bloom filters are kept in the LruBlockCache and the data blocks are kept in the BucketCache.
<code>hbase.bucketcache.ioengine</code>	none (BucketCache is disabled by default)	Where to store the contents of the BucketCache. Either <code>onheap</code> or <code>file:/path/to/file</code> .
<code>hfile.block.cache.size</code>	0.4	A float between 0.0 and 1.0. This factor multiplied by the Java heap size is the size of the L1 cache. In other words, the percentage of the Java heap to use for the L1 cache.
<code>hbase.bucketcache.size</code>	not set	When using BucketCache, this is a float that represents one of two different values , depending on whether it is a floating-point decimal less than 1.0 or an integer greater than 1.0. <ul style="list-style-type: none"> • If less than 1.0, it represents a percentage of total heap memory size to give to the cache. • If greater than 1.0, it represents the capacity of the cache in megabytes
<code>hbase.bucketcache.bucket.sizes</code>	4, 8, 16, 32, 40, 48, 56, 64, 96, 128, 192, 256, 384, 512 KB	A comma-separated list of sizes for buckets for the BucketCache if you prefer to use multiple sizes. The sizes should be multiples of the default blocksize, ordered from smallest to largest. The sizes you use will depend on your data patterns. This parameter is experimental.
<code>-XX:MaxDirectMemorySize</code>	<code>MaxDirectMemorySize = BucketCache + 1</code>	A JVM option to configure the maximum amount of direct memory available to the JVM. It is automatically calculated and configured based on the following formula: <code>MaxDirectMemorySize = BucketCache size + 1 GB</code> for other features using direct memory,

Property	Default	Description
		such as DFSClient. For example, if the BucketCache size is 8 GB, it will be <code>-XX:MaxDirectMemorySize=9G</code> .



1. 20% minimum reserved for operations and rpc call queues
2. `hbase.regionserver.global.memstore.size`: default is 0.4, which means 40%
3. `hbase.regionserver.global.memstore.size + hfile.block.cache.size` ≤ 0.80 , which means 80%
4. `hfile.block.cache.size`: default is 0.4, which means 40%
5. slack reserved for HDFS SCR/NIO: number of open HFiles * `hbase.dfs.client.read.shortcircuit.buffer.size`, where `hbase.dfs.client.read.shortcircuit.buffer.size` is set to 128k.

Figure 1: Default LRU Cache, L1 only block cache `hbase.bucketcache.ioengine=NULL`



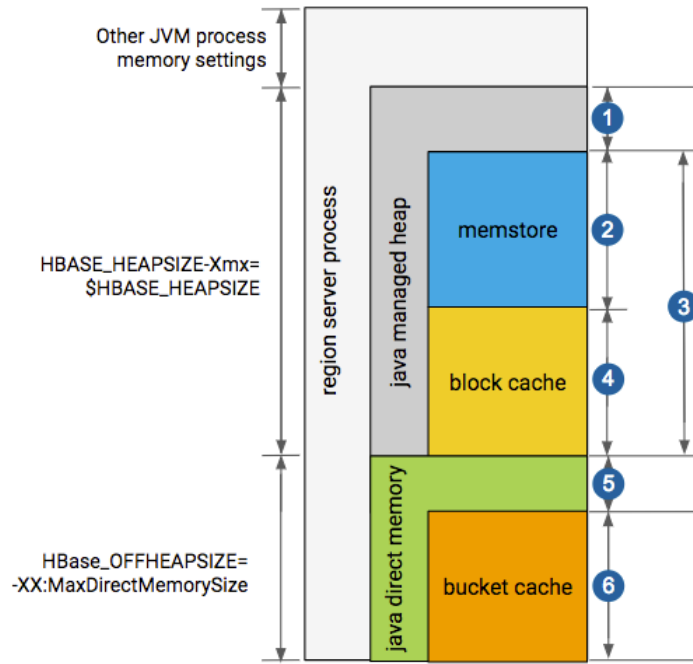
1. 20% minimum reserved for operations and rpc call queues
2. `hbase.regionserver.global.memstore.size`: default is 0.4, which means 40%
3. `hbase.regionserver.global.memstore.size + hfile.block.cache.size` ≤ 0.80 , which means 80%
4. `hfile.block.cache.size`: default is 0.4, which means 40%
5. slack reserved for HDFS SCR/NIO: number of open HFiles * `hbase.dfs.client.read.shortcircuit.buffer.size`, where `hbase.dfs.client.read.shortcircuit.buffer.size` is set to 128k.

6. `hbase.bucketcache.size`: default is 0.0

If `hbase.bucketcache.size` is float <1, it represents the percentage of total heap size.

If `hbase.bucketcache.size` is ≥ 1 , it represents the absolute value in MB. It must be <HBASE_OFFHEAPSIZE

Figure 2: Default LRU Cache, L1 only block cache `hbase.bucketcache.ioengine=heap`



1. 20% minimum reserved for operations and rpc call queues

2. `hbase.regionserver.global.memstore.size`: default is 0.4, which means 40%

3. `hbase.regionserver.global.memstore.size + hfile.block.cache.size` ≤ 0.80 , which means 80%

4. `hfile.block.cache.size`: default is 0.4 which means 40%

5. slack reserved for HDFS SCR/NIO: number of open HFiles * `hbase.dfs.client.read.shortcircuit.buffer.size`, where `hbase.dfs.client.read.shortcircuit.buffer.size` is set to 128k.

6. `hbase.bucketcache.size`: default is 0.0

If `hbase.bucketcache.size` is float <1, it represents the percentage of total heap size.

If `hbase.bucketcache.size` is ≥ 1 , it represents the absolute value in MB.

Figure 3: Default LRU Cache, L1 only block cache `hbase.bucketcache.ioengine=offheap`

Configuring the Off-heap BucketCache Using Cloudera Manager

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select the **RegionServer** scope and do the following:
 - a. Ensure that **Enable Combined BucketCache** is selected.
 - b. Set **BucketCache IOEngine** to `offheap`.
 - c. Update the value of **BucketCache Size** according to the required BucketCache size.
4. In the **HBase Region Server Environment Advanced Configuration Snippet for hbase-env.sh**, edit the following parameters:

- `HBASE_OFFHEAPSIZE`: Set it to a value (such as 5G) that accommodates your required L2 cache size, in addition to space reserved for cache management.
 - `HBASE_OPTS`: Add the JVM option `--XX:MaxDirectMemorySize=<size>G`, replacing `<size>` with a value not smaller than the aggregated heap size expressed as a number of gigabytes + the off-heap BucketCache, expressed as a number of gigabytes + around 1GB used for HDFS short circuit read. For example, if the off-heap BucketCache is 16GB and the heap size is 15GB, the total value of `MaxDirectMemorySize` could be 32:
`--XX:MaxDirectMemorySize=32G`.
5. Optionally, when combined BucketCache is in use, you can decrease the heap size ratio allocated to the L1 BlockCache, and increase the Memstore size.
 The on-heap BlockCache only stores indexes and Bloom filters, the actual data resides in the off-heap BucketCache. A larger Memstore is able to accommodate more write request before flushing them to disks.
 - Decrease **HFile Block Cache Size** to 0.3 or 0.2.
 - Increase **Maximum Size of All Memstores in RegionServer** to 0.5 or 0.6 respectively.
 6. Click **Save Changes** to commit the changes.
 7. Restart or rolling restart your RegionServers for the changes to take effect.

Configuring the Off-heap BucketCache Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Verify the RegionServer's off-heap size, and if necessary, tune it by editing the `hbase-env.sh` file and adding a line like the following:

```
HBASE_OFFHEAPSIZE=5G
```

Set it to a value that accommodates your BucketCache size + the additional space required for DFSClnt short circuit reads (default is 1 GB). For example, if the BucketCache size is 8 GB, set `HBASE_OFFHEAPSIZE=9G`.

2. Configure the `MaxDirectMemorySize` option for the RegionServers JVMs. This can be done adding the following line in `hbase-env.sh`:

```
HBASE_REGIONSERVER_OPTS="$HBASE_REGIONSERVER_OPTS
-XX:MaxDirectMemorySize=<size>G
```

Replace `<size>` with the same value set for `HBASE_OFFHEAPSIZE`.

3. Next, in the `hbase-site.xml` files on the RegionServers, configure the properties in [Table 1: BucketCache Configuration Properties](#) on page 90 as appropriate, using the example below as a model.

```
<property>
  <name>hbase.bucketcache.combinedcache.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.bucketcache.ioengine</name>
  <value>offheap</value>
</property>
<property>
  <name>hbase.bucketcache.size</name>
  <value>8388608</value>
</property>
<property>
  <name>hfile.block.cache.size</name>
```

```
<value>0.2</value>
</property>
<property>
<name>hbase.regionserver.global.memstore.size</name>
<value>0.6</value>
</property>
```

4. Restart each RegionServer for the changes to take effect.

Monitoring the BlockCache

Cloudera Manager provides metrics to monitor the performance of the BlockCache, to assist you in tuning your configuration.

You can view further detail and graphs using the RegionServer UI. To access the RegionServer UI in Cloudera Manager, go to the Cloudera Manager page for the host, click the **RegionServer** process, and click **HBase RegionServer Web UI**.

If you do not use Cloudera Manager, access the BlockCache reports at

`http://regionServer_host:22102/rs-status#memoryStats`, replacing `regionServer_host` with the hostname or IP address of your RegionServer.

Reading Data from HBase

[Get](#) and [Scan](#) are the two ways to read data from HBase, aside from manually parsing HFiles. A `Get` is simply a `Scan` limited by the API to one row. A `Scan` fetches zero or more rows of a table. By default, a `Scan` reads the entire table from start to end. You can limit your `Scan` results in several different ways, which affect the `Scan`'s load in terms of IO, network, or both, as well as processing load on the client side. This topic is provided as a quick reference. Refer to the [API documentation for Scan](#) for more in-depth information. You can also perform Gets and Scan using the HBase Shell.

- Specify a `startrow` or `stoprow` or both. Neither `startrow` nor `stoprow` need to exist. Because HBase sorts rows lexicographically, it will return the first row after `startrow` would have occurred, and will stop returning rows after `stoprow` would have occurred. The goal is to reduce IO and network.
 - The `startrow` is inclusive and the `stoprow` is exclusive. Given a table with rows `a`, `b`, `c`, `d`, `e`, `f`, and `startrow` of `c` and `stoprow` of `f`, rows `c-e` are returned.
 - If you omit `startrow`, the first row of the table is the `startrow`.
 - If you omit the `stoprow`, all results after `startrow` (including `startrow`) are returned.
 - If `startrow` is lexicographically after `stoprow`, and you set `Scan setReversed(boolean reversed)` to `true`, the results are returned in reverse order. Given the same table above, with rows `a-f`, if you specify `c` as the `stoprow` and `f` as the `startrow`, rows `f`, `e`, and `d` are returned.

```
Scan()
Scan(byte[] startRow)
Scan(byte[] startRow, byte[] stopRow)
```

- Specify a scanner cache that will be filled before the `Scan` result is returned, setting `setCaching` to the number of rows to cache before returning the result. By default, the caching setting on the table is used. The goal is to balance IO and network load.

```
public Scan setCaching(int caching)
```

- To limit the number of columns if your table has very wide rows (rows with a large number of columns), use `setBatch(int batch)` and set it to the number of columns you want to return in one batch. A large number of columns is not a recommended design pattern.

```
public Scan setBatch(int batch)
```

- To specify a maximum result size, use `setMaxResultSize(long)`, with the number of bytes. The goal is to reduce IO and network.

```
public Scan setMaxResultSize(long maxResultSize)
```

- When you use `setCaching` and `setMaxResultSize` together, single server requests are limited by either number of rows or maximum result size, whichever limit comes first.
- You can limit the scan to specific column families or columns by using `addFamily` or `addColumn`. The goal is to reduce IO and network. IO is reduced because each column family is represented by a Store on each RegionServer, and only the Stores representing the specific column families in question need to be accessed.

```
public Scan addColumn(byte[] family,
                    byte[] qualifier)

public Scan addFamily(byte[] family)
```

- You can specify a range of timestamps or a single timestamp by specifying `setTimeRange` or `setTimeStamp`.

```
public Scan setTimeRange(long minStamp,
                       long maxStamp)
                       throws IOException

public Scan setTimeStamp(long timestamp)
                       throws IOException
```

- You can retrieve a maximum number of versions by using `setMaxVersions`.

```
public Scan setMaxVersions(int maxVersions)
```

- You can use a filter by using `setFilter`. Filters are discussed in detail in [HBase Filtering](#) on page 96 and the [Filter API](#).

```
public Scan setFilter(Filter filter)
```

- You can disable the server-side block cache for a specific scan using the API `setCacheBlocks(boolean)`. This is an expert setting and should only be used if you know what you are doing.

Perform Scans Using HBase Shell

You can perform scans using HBase Shell, for testing or quick queries. Use the following guidelines or issue the scan command in HBase Shell with no parameters for more usage information. This represents only a subset of possibilities.

```
# Display usage information
hbase> scan

# Scan all rows of table 't1'
hbase> scan 't1'

# Specify a startrow, limit the result to 10 rows, and only return selected columns
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], LIMIT => 10, STARTROW => 'xyz'}

# Specify a timerange
hbase> scan 't1', {TIMERANGE => [1303668804, 1303668904]}

# Specify a custom filter
hbase> scan 't1', {FILTER => org.apache.hadoop.hbase.filter.ColumnPaginationFilter.new(1,
0)}

# Disable the block cache for a specific scan (experts only)
hbase> scan 't1', {COLUMNS => ['c1', 'c2'], CACHE_BLOCKS => false}
```

Hedged Reads

Hadoop 2.4 introduced a new feature called *hedged reads*. If a read from a block is slow, the HDFS client starts up another parallel, 'hedged' read against a different block replica. The result of whichever read returns first is used, and the outstanding read is cancelled. This feature helps in situations where a read occasionally takes a long time rather than when there is a systemic problem. Hedged reads can be enabled for HBase when the HFiles are stored in HDFS. This feature is disabled by default.

Enabling Hedged Reads for HBase Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

To enable hedged reads for HBase, edit the `hbase-site.xml` file on each server. Set `dfs.client.hedged.read.threadpool.size` to the number of threads to dedicate to running hedged threads, and set the `dfs.client.hedged.read.threshold.millis` configuration property to the number of milliseconds to wait before starting a second read against a different block replica. Set `dfs.client.hedged.read.threadpool.size` to 0 or remove it from the configuration to disable the feature. After changing these properties, restart your cluster.

The following is an example configuration for hedged reads for HBase.

```
<property>
  <name>dfs.client.hedged.read.threadpool.size</name>
  <value>20</value>  <!-- 20 threads -->
</property>
<property>
  <name>dfs.client.hedged.read.threshold.millis</name>
  <value>10</value>  <!-- 10 milliseconds -->
</property>
```

HBase Filtering

When reading data from HBase using Get or Scan operations, you can use custom filters to return a subset of results to the client. While this does not reduce server-side IO, it does reduce network bandwidth and reduces the amount of data the client needs to process. Filters are generally used using the Java API, but can be used from HBase Shell for testing and debugging purposes.

For more information on Gets and Scans in HBase, see [Reading Data from HBase](#) on page 94.

Filter Syntax Guidelines

HBase filters take zero or more arguments, in parentheses. Where the argument is a string, it is surrounded by single quotes ('string').

Logical Operators, Comparison Operators and Comparators

Filters can be combined together with logical operators. Some filters take a combination of comparison operators and comparators. Following is the list of each.

Logical Operators

- AND - the key-value must pass both the filters to be included in the results.
- OR - the key-value must pass at least one of the filters to be included in the results.
- SKIP - for a particular row, if any of the key-values do not pass the filter condition, the entire row is skipped.
- WHILE - For a particular row, it continues to emit key-values until a key-value is reached that fails the filter condition.

- Compound Filters - Using these operators, a hierarchy of filters can be created. For example:

```
(Filter1 AND Filter2)OR(Filter3 AND Filter4)
```

Comparison Operators

- LESS (<)
- LESS_OR_EQUAL (<=)
- EQUAL (=)
- NOT_EQUAL (!=)
- GREATER_OR_EQUAL (>=)
- GREATER (>)
- NO_OP (no operation)

Comparators

- BinaryComparator - lexicographically compares against the specified byte array using the `Bytes.compareTo(byte[], byte[])` method.
- BinaryPrefixComparator - lexicographically compares against a specified byte array. It only compares up to the length of this byte array.
- RegexStringComparator - compares against the specified byte array using the given regular expression. Only `EQUAL` and `NOT_EQUAL` comparisons are valid with this comparator.
- SubStringComparator - tests whether or not the given substring appears in a specified byte array. The comparison is case insensitive. Only `EQUAL` and `NOT_EQUAL` comparisons are valid with this comparator.

Examples

```
Example1: >, 'binary:abc' will match everything that is lexicographically greater than "abc"
Example2: =, 'binaryprefix:abc' will match everything whose first 3 characters are lexicographically equal to "abc"
Example3: !=, 'regexstring:ab*yz' will match everything that doesn't begin with "ab" and ends with "yz"
Example4: =, 'substring:abc123' will match everything that begins with the substring "abc123"
```

Compound Operators

Within an expression, parentheses can be used to group clauses together, and parentheses have the highest order of precedence.

`SKIP` and `WHILE` operators are next, and have the same precedence.

The `AND` operator is next.

The `OR` operator is next.

Examples

```
A filter string of the form: "Filter1 AND Filter2 OR Filter3" will be evaluated as: "(Filter1 AND Filter2) OR Filter3"
A filter string of the form: "Filter1 AND SKIP Filter2 OR Filter3" will be evaluated as: "(Filter1 AND (SKIP Filter2)) OR Filter3"
```

Filter Types

HBase includes several filter types, as well as the ability to group filters together and create your own custom filters.

- **KeyOnlyFilter** - takes no arguments. Returns the key portion of each key-value pair.

```
Syntax: KeyOnlyFilter ()
```

- **FirstKeyOnlyFilter** - takes no arguments. Returns the key portion of the first key-value pair.

```
Syntax: FirstKeyOnlyFilter ()
```

- **PrefixFilter** - takes a single argument, a prefix of a row key. It returns only those key-values present in a row that start with the specified row prefix

```
Syntax: PrefixFilter (<row_prefix>)
```

```
Example: PrefixFilter ('Row')
```

- **ColumnPrefixFilter** - takes a single argument, a column prefix. It returns only those key-values present in a column that starts with the specified column prefix.

```
Syntax: ColumnPrefixFilter (<column_prefix>)
```

```
Example: ColumnPrefixFilter ('Col')
```

- **MultipleColumnPrefixFilter** - takes a list of column prefixes. It returns key-values that are present in a column that starts with *any* of the specified column prefixes.

```
Syntax: MultipleColumnPrefixFilter (<column_prefix>, <column_prefix>, ..., <column_prefix>)
```

```
Example: MultipleColumnPrefixFilter ('Col1', 'Col2')
```

- **ColumnCountGetFilter** - takes one argument, a limit. It returns the first limit number of columns in the table.

```
Syntax: ColumnCountGetFilter (<limit>)
```

```
Example: ColumnCountGetFilter (4)
```

- **PageFilter** - takes one argument, a page size. It returns page size number of rows from the table.

```
Syntax: PageFilter (<page_size>)
```

```
Example: PageFilter (2)
```

- **ColumnPaginationFilter** - takes two arguments, a limit and offset. It returns limit number of columns after offset number of columns. It does this for all the rows.

```
Syntax: ColumnPaginationFilter (<limit>, <offset>)
```

```
Example: ColumnPaginationFilter (3, 5)
```

- **InclusiveStopFilter** - takes one argument, a row key on which to stop scanning. It returns all key-values present in rows *up to and including* the specified row.

```
Syntax: InclusiveStopFilter (<stop_row_key>)
```

```
Example: InclusiveStopFilter ('Row2')
```

- **TimeStampsFilter** - takes a list of timestamps. It returns those key-values whose timestamps matches *any* of the specified timestamps.

```
Syntax: TimeStampsFilter (<timestamp>, <timestamp>, ... ,<timestamp>)
```

```
Example: TimeStampsFilter (5985489, 48895495, 58489845945)
```

- **RowFilter** - takes a compare operator and a comparator. It compares each row key with the comparator using the compare operator and if the comparison returns `true`, it returns all the key-values in that row.

```
Syntax: RowFilter (<compareOp>, '<row_comparator>')
```

```
Example: RowFilter (<=>, 'binary:xyz')
```

- **FamilyFilter** - takes a compare operator and a comparator. It compares each family name with the comparator using the compare operator and if the comparison returns `true`, it returns all the key-values in that family.

```
Syntax: FamilyFilter (<compareOp>, '<family_comparator>')
```

```
Example: FamilyFilter (>=>, 'binaryprefix:FamilyB')
```

- **QualifierFilter** - takes a compare operator and a comparator. It compares each qualifier name with the comparator using the compare operator and if the comparison returns `true`, it returns all the key-values in that column.

```
Syntax: QualifierFilter (<compareOp>, '<qualifier_comparator>')
```

```
Example: QualifierFilter (=, 'substring:Column1')
```

- **ValueFilter** - takes a compare operator and a comparator. It compares each value with the comparator using the compare operator and if the comparison returns `true`, it returns that key-value.

```
Syntax: ValueFilter (<compareOp>, '<value_comparator>')
```

```
Example: ValueFilter (!=>, 'binary:Value')
```

- **DependentColumnFilter** - takes two arguments required arguments, a family and a qualifier. It tries to locate this column in each row and returns all key-values in that row that have the same timestamp. If the row does not contain the specified column, none of the key-values in that row will be returned.

The filter can also take an optional boolean argument, `dropDependentColumn`. If set to `true`, the column used for the filter does not get returned.

The filter can also take two more additional optional arguments, a compare operator and a value comparator, which are further checks in addition to the family and qualifier. If the dependent column is found, its value should also pass the value check. If it does pass the value check, only then is its timestamp taken into consideration.

```
Syntax: DependentColumnFilter ('<family>', '<qualifier>', <boolean>, <compare operator>,
'<value comparator>')
DependentColumnFilter ('<family>', '<qualifier>', <boolean>)
DependentColumnFilter ('<family>', '<qualifier>')
```

```
Example: DependentColumnFilter ('conf', 'blacklist', false, >=>, 'zebra')
DependentColumnFilter ('conf', 'blacklist', true)
DependentColumnFilter ('conf', 'blacklist')
```

- **SingleColumnValueFilter** - takes a column family, a qualifier, a compare operator and a comparator. If the specified column is not found, all the columns of that row will be emitted. If the column is found and the comparison with the comparator returns `true`, all the columns of the row will be emitted. If the condition fails, the row will not be emitted.

This filter also takes two additional optional boolean arguments, `filterIfColumnMissing` and `setLatestVersionOnly`.

If the `filterIfColumnMissing` flag is set to `true`, the columns of the row will not be emitted if the specified column to check is not found in the row. The default value is `false`.

If the `setLatestVersionOnly` flag is set to `false`, it will test previous versions (timestamps) in addition to the most recent. The default value is `true`.

These flags are optional and dependent on each other. You must set neither or both of them together.

```
Syntax: SingleColumnValueFilter (<family>', '<qualifier>', <compare operator>,
'<comparator>', <filterIfColumnMissing_boolean>, <latest_version_boolean>)
Syntax: SingleColumnValueFilter (<family>', '<qualifier>', <compare operator>,
'<comparator>')
Example: SingleColumnValueFilter ('FamilyA', 'Column1', <=, 'abc', true, false)
Example: SingleColumnValueFilter ('FamilyA', 'Column1', <=, 'abc')
```

- **SingleColumnValueExcludeFilter** - takes the same arguments and behaves same as SingleColumnValueFilter. However, if the column is found and the condition passes, all the columns of the row will be emitted except for the tested column value.

```
Syntax: SingleColumnValueExcludeFilter (<family>, <qualifier>, <compare operators>,
<comparator>, <latest_version_boolean>, <filterIfColumnMissing_boolean>)
Syntax: SingleColumnValueExcludeFilter (<family>, <qualifier>, <compare operator>
<comparator>)
Example: SingleColumnValueExcludeFilter ('FamilyA', 'Column1', '<=', 'abc', 'false',
'true')
Example: SingleColumnValueExcludeFilter ('FamilyA', 'Column1', '<=', 'abc')
```

- **ColumnRangeFilter** - takes either minColumn, maxColumn, or both. Returns only those keys with columns that are between minColumn and maxColumn. It also takes two boolean variables to indicate whether to include the minColumn and maxColumn or not. If you don't want to set the minColumn or the maxColumn, you can pass in an empty argument.

```
Syntax: ColumnRangeFilter (<minColumn >', <minColumnInclusive_bool>, '<maxColumn>',
<maxColumnInclusive_bool>)
Example: ColumnRangeFilter ('abc', true, 'xyz', false)
```

- **Custom Filter** - You can create a custom filter by implementing the [Filter](#) class. The JAR must be available on all RegionServers.

HBase Shell Example

This example scans the 'users' table for rows where the contents of the cf:name column equals the string 'abc'.

```
hbase> scan 'users', { FILTER => SingleColumnValueFilter.new(Bytes.toBytes('cf'),
Bytes.toBytes('name'), CompareFilter::CompareOp.valueOf('EQUAL'),
BinaryComparator.new(Bytes.toBytes('abc')))}
```

Java API Example

This example, taken from the HBase unit test found in

hbase-server/src/test/java/org/apache/hadoop/hbase/filter/TestSingleColumnValueFilter.java, shows how to use the Java API to implement several different filters..

```
/**
 *
 * Licensed to the Apache Software Foundation (ASF) under one
 * or more contributor license agreements. See the NOTICE file
 * distributed with this work for additional information
 * regarding copyright ownership. The ASF licenses this file
 * to you 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.
*/
package org.apache.hadoop.hbase.filter;

import static org.junit.Assert.assertFalse;
import static org.junit.Assert.assertTrue;

import java.util.regex.Pattern;

import org.apache.hadoop.hbase.KeyValue;
import org.apache.hadoop.hbase.SmallTests;
import org.apache.hadoop.hbase.filter.CompareFilter.CompareOp;
import org.apache.hadoop.hbase.util.Bytes;
import org.junit.Before;
import org.junit.Test;
import org.junit.experimental.categories.Category;

/**
 * Tests the value filter
 */
@Category(SmallTests.class)
public class TestSingleColumnValueFilter {
    private static final byte[] ROW = Bytes.toBytes("test");
    private static final byte[] COLUMN_FAMILY = Bytes.toBytes("test");
    private static final byte[] COLUMN_QUALIFIER = Bytes.toBytes("foo");
    private static final byte[] VAL_1 = Bytes.toBytes("a");
    private static final byte[] VAL_2 = Bytes.toBytes("ab");
    private static final byte[] VAL_3 = Bytes.toBytes("abc");
    private static final byte[] VAL_4 = Bytes.toBytes("abcd");
    private static final byte[] FULLSTRING_1 =
        Bytes.toBytes("The quick brown fox jumps over the lazy dog.");
    private static final byte[] FULLSTRING_2 =
        Bytes.toBytes("The slow grey fox trips over the lazy dog.");
    private static final String QUICK_SUBSTR = "quick";
    private static final String QUICK_REGEX = ".+quick.+";
    private static final Pattern QUICK_PATTERN = Pattern.compile("QuIcK",
        Pattern.CASE_INSENSITIVE | Pattern.DOTALL);

    Filter basicFilter;
    Filter nullFilter;
    Filter substrFilter;
    Filter regexFilter;
    Filter regexPatternFilter;

    @Before
    public void setUp() throws Exception {
        basicFilter = basicFilterNew();
        nullFilter = nullFilterNew();
        substrFilter = substrFilterNew();
        regexFilter = regexFilterNew();
        regexPatternFilter = regexFilterNew(QUICK_PATTERN);
    }

    private Filter basicFilterNew() {
        return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
            CompareOp.GREATER_OR_EQUAL, VAL_2);
    }

    private Filter nullFilterNew() {
        return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
            CompareOp.NOT_EQUAL,
            new NullComparator());
    }

    private Filter substrFilterNew() {
        return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
            CompareOp.EQUAL,
            new SubstringComparator(QUICK_SUBSTR));
    }

    private Filter regexFilterNew() {

```

```

        return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
            CompareOp.EQUAL,
            new RegexStringComparator(QUICK_REGEX));
    }

    private Filter regexFilterNew(Pattern pattern) {
        return new SingleColumnValueFilter(COLUMN_FAMILY, COLUMN_QUALIFIER,
            CompareOp.EQUAL,
            new RegexStringComparator(pattern.pattern(), pattern.flags()));
    }

    private void basicFilterTests(SingleColumnValueFilter filter)
        throws Exception {
        KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_2);
        assertTrue("basicFilter1", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);

        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_3);
        assertTrue("basicFilter2", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);

        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_4);
        assertTrue("basicFilter3", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);

        assertFalse("basicFilterNotNull", filter.filterRow());
        filter.reset();
        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_1);
        assertTrue("basicFilter4", filter.filterKeyValue(kv) == Filter.ReturnCode.NEXT_ROW);

        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_2);
        assertTrue("basicFilter4", filter.filterKeyValue(kv) == Filter.ReturnCode.NEXT_ROW);

        assertFalse("basicFilterAllRemaining", filter.filterAllRemaining());
        assertTrue("basicFilterNotNull", filter.filterRow());
        filter.reset();
        filter.setLatestVersionOnly(false);
        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_1);
        assertTrue("basicFilter5", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);

        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, VAL_2);
        assertTrue("basicFilter5", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);

        assertFalse("basicFilterNotNull", filter.filterRow());
    }

    private void nullFilterTests(Filter filter) throws Exception {
        ((SingleColumnValueFilter) filter).setFilterIfMissing(true);
        KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER, FULLSTRING_1);
        assertTrue("null1", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
        assertFalse("null1FilterRow", filter.filterRow());
        filter.reset();
        kv = new KeyValue(ROW, COLUMN_FAMILY, Bytes.toBytes("qual2"), FULLSTRING_2);
        assertTrue("null2", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
        assertTrue("null2FilterRow", filter.filterRow());
    }

    private void substrFilterTests(Filter filter)
        throws Exception {
        KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
            FULLSTRING_1);
        assertTrue("substrTrue",
            filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
        kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
            FULLSTRING_2);
        assertTrue("substrFalse", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
        assertFalse("substrFilterAllRemaining", filter.filterAllRemaining());
        assertFalse("substrFilterNotNull", filter.filterRow());
    }

    private void regexFilterTests(Filter filter)
        throws Exception {
        KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
            FULLSTRING_1);
        assertTrue("regexTrue",
            filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
    }

```

```

kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
    FULLSTRING_2);
assertTrue("regexFalse", filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
assertFalse("regexFilterAllRemaining", filter.filterAllRemaining());
assertFalse("regexFilterNotNull", filter.filterRow());
}

private void regexPatternFilterTests(Filter filter)
    throws Exception {
    KeyValue kv = new KeyValue(ROW, COLUMN_FAMILY, COLUMN_QUALIFIER,
        FULLSTRING_1);
    assertTrue("regexTrue",
        filter.filterKeyValue(kv) == Filter.ReturnCode.INCLUDE);
    assertFalse("regexFilterAllRemaining", filter.filterAllRemaining());
    assertFalse("regexFilterNotNull", filter.filterRow());
}

private Filter serializationTest(Filter filter)
    throws Exception {
    // Decompose filter to bytes.
    byte[] buffer = filter.toByteArray();

    // Recompose filter.
    Filter newFilter = SingleColumnValueFilter.parseFrom(buffer);
    return newFilter;
}

/**
 * Tests identification of the stop row
 * @throws Exception
 */
@Test
public void testStop() throws Exception {
    basicFilterTests((SingleColumnValueFilter) basicFilter);
    nullFilterTests(nullFilter);
    substrFilterTests(substrFilter);
    regexFilterTests(regexFilter);
    regexPatternFilterTests(regexPatternFilter);
}

/**
 * Tests serialization
 * @throws Exception
 */
@Test
public void testSerialization() throws Exception {
    Filter newFilter = serializationTest(basicFilter);
    basicFilterTests((SingleColumnValueFilter)newFilter);
    newFilter = serializationTest(nullFilter);
    nullFilterTests(newFilter);
    newFilter = serializationTest(substrFilter);
    substrFilterTests(newFilter);
    newFilter = serializationTest(regexFilter);
    regexFilterTests(newFilter);
    newFilter = serializationTest(regexPatternFilter);
    regexPatternFilterTests(newFilter);
}
}

```

Writing Data to HBase

To write data to HBase, you use methods of the `HTableInterface` class. You can use the Java API directly, or use HBase Shell, Thrift API, REST API, or another client which uses the Java API indirectly. When you issue a Put, the coordinates of the data are the row, the column, and the timestamp. The timestamp is unique per version of the cell, and can be generated automatically or specified programmatically by your application, and must be a long integer.

Variations on Put

There are several different ways to write data into HBase. Some of them are listed below.

- A `Put` operation writes data into HBase.
- A `Delete` operation deletes data from HBase. What actually happens during a `Delete` depends upon several factors.
- A `CheckAndPut` operation performs a `Scan` before attempting the `Put`, and only does the `Put` if a value matches what is expected, and provides row-level atomicity.
- A `CheckAndDelete` operation performs a `Scan` before attempting the `Delete`, and only does the `Delete` if a value matches what is expected.
- An `Increment` operation increments values of one or more columns within a single row, and provides row-level atomicity.

Refer to the API documentation for a full list of methods provided for writing data to HBase. Different methods require different access levels and have other differences.

Versions

When you put data into HBase, a timestamp is required. The timestamp can be generated automatically by the `RegionServer` or can be supplied by you. The timestamp must be unique per version of a given cell, because the timestamp identifies the version. To modify a previous version of a cell, for instance, you would issue a `Put` with a different value for the data itself, but the same timestamp.

HBase's behavior regarding versions is highly configurable. The maximum number of versions defaults to 1 in CDH 5, and 3 in previous versions. You can change the default value for HBase by configuring `hbase.column.max.version` in `hbase-site.xml`, either using an advanced configuration snippet if you use Cloudera Manager, or by editing the file directly otherwise.

You can also configure the maximum and minimum number of versions to keep for a given column, or specify a default time-to-live (TTL), which is the number of seconds before a version is deleted. The following examples all use `alter` statements in HBase Shell to create new column families with the given characteristics, but you can use the same syntax when creating a new table or to alter an existing column family. This is only a fraction of the options you can specify for a given column family.

```
hbase> alter 't1', NAME => 'f1', VERSIONS => 5
hbase> alter 't1', NAME => 'f1', MIN_VERSIONS => 2
hbase> alter 't1', NAME => 'f1', TTL => 15
```

HBase sorts the versions of a cell from newest to oldest, by sorting the timestamps lexicographically. When a version needs to be deleted because a threshold has been reached, HBase always chooses the "oldest" version, even if it is in fact the most recent version to be inserted. Keep this in mind when designing your timestamps. Consider using the default generated timestamps and storing other version-specific data elsewhere in the row, such as in the row key. If `MIN_VERSIONS` and `TTL` conflict, `MIN_VERSIONS` takes precedence.

Deletion

When you request for HBase to delete data, either explicitly using a `Delete` method or implicitly using a threshold such as the maximum number of versions or the TTL, HBase does not delete the data immediately. Instead, it writes a deletion marker, called a tombstone, to the HFile, which is the physical file where a given `RegionServer` stores its region of a column family. The tombstone markers are processed during major compaction operations, when HFiles are rewritten without the deleted data included.

Even after major compactions, "deleted" data may not actually be deleted. You can specify the `KEEP_DELETED_CELLS` option for a given column family, and the tombstones will be preserved in the HFile even after major compaction. One scenario where this approach might be useful is for data retention policies.

Another reason deleted data may not actually be deleted is if the data would be required to restore a table from a snapshot which has not been deleted. In this case, the data is moved to an archive during a major compaction, and only deleted when the snapshot is deleted. This is a good reason to monitor the number of snapshots saved in HBase.

Examples

This abbreviated example writes data to an HBase table using HBase Shell and then scans the table to show the result.

```
hbase> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.1770 seconds

hbase> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0160 seconds

hbase> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0260 seconds
hbase> scan 'test'
ROW                                COLUMN+CELL
 row1                                column=cf:a, timestamp=1403759475114, value=value1
 row2                                column=cf:b, timestamp=1403759492807, value=value2
 row3                                column=cf:c, timestamp=1403759503155, value=value3
3 row(s) in 0.0440 seconds
```

This abbreviated example uses the HBase API to write data to an HBase table, using the automatic timestamp created by the Region Server.

```
publicstaticfinalbyte[] CF = "cf".getBytes();
publicstaticfinalbyte[] ATTR = "attr".getBytes();
...
Put put = new Put(Bytes.toBytes(row));
put.add(CF, ATTR, Bytes.toBytes(data));
htable.put(put);
```

This example uses the HBase API to write data to an HBase table, specifying the timestamp.

```
publicstaticfinalbyte[] CF = "cf".getBytes();
publicstaticfinalbyte[] ATTR = "attr".getBytes();
...
Put put = new Put(Bytes.toBytes(row));
long explicitTimeInMs = 555; // just an example
put.add(CF, ATTR, explicitTimeInMs, Bytes.toBytes(data));
htable.put(put);
```

Further Reading

- Refer to the [HTableInterface](#) and [HColumnDescriptor](#) API documentation for more details about configuring tables and columns, as well as reading and writing to HBase.
- Refer to the [Apache HBase Reference Guide](#) for more in-depth information about HBase, including details about versions and deletions not covered here.

Importing Data Into HBase

The method you use for importing data into HBase depends on several factors:

- The location, size, and format of your existing data
- Whether you need to import data once or periodically over time
- Whether you want to import the data in bulk or stream it into HBase regularly
- How fresh the HBase data needs to be

This topic helps you choose the correct method or composite of methods and provides example workflows for each method.

Always run HBase administrative commands as the HBase user (typically `hbase`).

Choosing the Right Import Method

If the data is already in an HBase table:

- To move the data from one HBase cluster to another, use `snapshot` and either the `clone_snapshot` or `ExportSnapshot` utility; or, use the `CopyTable` utility.

- To move the data from one HBase cluster to another without downtime on either cluster, use replication.
- To migrate data between HBase version that are not wire compatible, such as from CDH 4 to CDH 5, see [Importing HBase Data From CDH 4 to CDH 5](#) on page 106.

If the data currently exists outside HBase:

- If possible, write the data to HFile format, and use a BulkLoad to import it into HBase. The data is immediately available to HBase and you can bypass the normal write path, increasing efficiency.
- If you prefer not to use bulk loads, and you are using a tool such as Pig, you can use it to import your data.

If you need to stream live data to HBase instead of import in bulk:

- Write a Java client using the Java API, or use the Apache Thrift Proxy API to write a client in a language supported by Thrift.
- Stream data directly into HBase using the REST Proxy API in conjunction with an HTTP client such as `wget` or `curl`.
- Use Flume or Spark.

Most likely, at least one of these methods works in your situation. If not, you can use MapReduce directly. Test the most feasible methods with a subset of your data to determine which one is optimal.

Using CopyTable

`CopyTable` uses HBase read and write paths to copy part or all of a table to a new table in either the same cluster or a different cluster. `CopyTable` causes read load when reading from the source, and write load when writing to the destination. Region splits occur on the destination table in real time as needed. To avoid these issues, use `snapshot` and `export` commands instead of `CopyTable`. Alternatively, you can pre-split the destination table to avoid excessive splits. The destination table can be partitioned differently from the source table. See [this section](#) of the Apache HBase documentation for more information.

Edits to the source table after the `CopyTable` starts are not copied, so you may need to do an additional `CopyTable` operation to copy new data into the destination table. Run `CopyTable` as follows, using `--help` to see details about possible parameters.

```
$ ./bin/hbase org.apache.hadoop.hbase.mapreduce.CopyTable --help
Usage: CopyTable [general options] [--starttime=X] [--endtime=Y] [--new.name=NEW]
[--peer.adr=ADR] <tablename>
```

The `starttime/endtime` and `startrow/endrow` pairs function in a similar way: if you leave out the first of the pair, the first timestamp or row in the table is the starting point. Similarly, if you leave out the second of the pair, the operation continues until the end of the table. To copy the table to a new table in the same cluster, you must specify `--new.name`, unless you want to write the copy back to the same table, which would add a new version of each cell (with the same data), or just overwrite the cell with the same value if the maximum number of versions is set to 1 (the default in CDH 5). To copy the table to a new table in a different cluster, specify `--peer.adr` and optionally, specify a new table name.

The following example creates a new table using HBase Shell in non-interactive mode, and then copies data in two ColumnFamilies in rows starting with timestamp 1265875194289 and including the last row before the `CopyTable` started, to the new table.

```
$ echo create 'NewTestTable', 'cf1', 'cf2', 'cf3' | bin/hbase shell --non-interactive
$ bin/hbase org.apache.hadoop.hbase.mapreduce.CopyTable --starttime=1265875194289
--families=cf1,cf2,cf3 --new.name=NewTestTable TestTable
```

In CDH 5, snapshots are recommended instead of `CopyTable` for most situations.

Importing HBase Data From CDH 4 to CDH 5

CDH 4 and CDH 5 are not wire-compatible, so import methods such as [CopyTable](#) will not work. Instead, you can use separate export and import operations using `distcp`, or you can copy the table's HFiles using HDFS utilities and upgrade

the HFiles in place. The first option is preferred unless the size of the table is too large to be practical and the export or import will take too long. The import/export mechanism gives you flexibility and allows you to run exports as often as you need, for an ongoing period of time. This would allow you to test CDH 5 with your production data before finalizing your upgrade, for instance.

Import and Export Data Using DistCP

1. Both Import and Export applications have several command-line options which you can use to control their behavior, such as limiting the import or export to certain column families or modifying the output directory. Run the commands without arguments to view the usage instructions. The output below is an example, and may be different for different HBase versions.

```
$ bin/hbase org.apache.hadoop.hbase.mapreduce.Import

Usage: Import [options] <tablename> <inputdir>
By default Import will load data directly into HBase. To instead generate
HFiles of data to prepare for a bulk data load, pass the option:
  -Dimport.bulk.output=/path/for/output
To apply a generic org.apache.hadoop.hbase.filter.Filter to the input, use
  -Dimport.filter.class=<name of filter class>
  -Dimport.filter.args=<comma separated list of args for filter
NOTE: The filter will be applied BEFORE doing key renames using the
HBASE_IMPORTER_RENAME_CFS property. Further, filters will only use the
  Filter#filterRowKey(byte[] buffer, int offset, int length) method to identify
whether the current row needs to be ignored completely
  for processing and Filter#filterKeyValue(KeyValue) method to determine if the
KeyValue should be added; Filter.ReturnCode#INCLUDE
  and #INCLUDE_AND_NEXT_COL will be considered as including the KeyValue.
To import data exported from HBase 0.94, use
  -Dhbase.import.version=0.94
For performance consider the following options:
  -Dmapreduce.map.speculative=false
  -Dmapreduce.reduce.speculative=false
  -Dimport.wal.durability=<Used while writing data to hbase. Allowed values
are the supported durability values like SKIP_WAL/ASYNC_WAL/SYNC_WAL/...>
```

```
$ /usr/bin/hbase org.apache.hadoop.hbase.mapreduce.Export

ERROR: Wrong number of arguments: 0
Usage: Export [-D <property=value>]* <tablename> <outputdir> [<versions> [<starttime>
[<endtime>]] [<regex pattern> or [<Prefix> to filter]]

Note: -D properties will be applied to the conf used.
For example:
  -D mapreduce.output.fileoutputformat.compress=true
  -D
mapreduce.output.fileoutputformat.compress.codec=org.apache.hadoop.io.compress.GzipCodec

  -D mapreduce.output.fileoutputformat.compress.type=BLOCK
Additionally, the following SCAN properties can be specified
to control/limit what is exported..
  -D hbase.mapreduce.scan.column.family=<familyName>
  -D hbase.mapreduce.include.deleted.rows=true
  -D hbase.mapreduce.scan.row.start=<ROWSTART>
  -D hbase.mapreduce.scan.row.stop=<ROWSTOP>
For performance consider the following properties:
  -Dhbase.client.scanner.caching=100
  -Dmapreduce.map.speculative=false
  -Dmapreduce.reduce.speculative=false
For tables with very wide rows consider setting the batch size as below:
  -Dhbase.export.scanner.batch=10
```

2. On the CDH 4 cluster, export the contents of the table to sequence files in a given directory using a command like the following.

```
$ sudo -u hdfs hbase org.apache.hadoop.hbase.mapreduce.Export <tablename>
/export_directory
```

The sequence files are located in the `/export_directory` directory.

3. Copy the contents of `/export_directory` to the CDH 5 cluster using `distcp` or through a filesystem accessible from hosts on both clusters. If you use `distcp`, the following is an example command.

```
$ sudo -u hdfs hadoop distcp -p -update -skipcrccheck  
hftp://cdh4-namenode:port/export_directory hdfs://cdh5-namenode/import_directory
```

4. Create the table on the CDH 5 cluster using HBase Shell. Column families must be identical to the table on the CDH 4 cluster.
5. Import the sequence file into the newly-created table.

```
$ sudo -u hdfs hbase -Dhbase.import.version=0.94 org.apache.hadoop.hbase.mapreduce.Import  
t1 /import_directory
```

Copy and Upgrade the HFiles

If exporting and importing the data is not feasible because of the size of the data or other reasons, or you know that the import will be a one-time occurrence, you can copy the HFiles directly from the CDH 4 cluster's HDFS filesystem to the CDH 5 cluster's HDFS filesystem, and upgrade the HFiles in place.



Warning: Only use this procedure if the destination cluster is a brand new HBase cluster with empty tables, and is not currently hosting any data. If this is not the case, or if you are unsure, contact Cloudera Support before following this procedure.

1. Use the `distcp` command on the CDH 5 cluster to copy the HFiles from the CDH 4 cluster.

```
$ sudo -u hdfs hadoop distcp -p -update -skipcrccheck  
webhdfs://cdh4-namenode:http-port/hbase hdfs://cdh5-namenode:rpc-port/hbase
```

2. In the destination cluster, upgrade the HBase tables. In Cloudera Manager, go to **Cluster** > **HBase** and choose **Upgrade HBase** from the **Action** menu. This will first check that the HBase tables are able to be upgraded, then upgrade them.
3. Start HBase on the CDH 5 cluster. The upgraded tables are available. Verify the data and confirm that no errors are logged.

Using Snapshots

As of CDH 4.7, Cloudera recommends snapshots instead of CopyTable where possible. A snapshot captures the state of a table at the time the snapshot was taken. Because no data is copied when a snapshot is taken, the process is very quick. As long as the snapshot exists, cells in the snapshot are never deleted from HBase, even if they are explicitly deleted by the API. Instead, they are archived so that the snapshot can restore the table to its state at the time of the snapshot.

After taking a snapshot, use the `clone_snapshot` command to copy the data to a new (immediately enabled) table in the same cluster, or the Export utility to create a new table based on the snapshot, in the same cluster or a new cluster. This is a copy-on-write operation. The new table shares HFiles with the original table until writes occur in the new table but not the old table, or until a compaction or split occurs in either of the tables. This can improve performance in the short term compared to CopyTable.

To export the snapshot to a new cluster, use the `ExportSnapshot` utility, which uses MapReduce to copy the snapshot to the new cluster. Run the `ExportSnapshot` utility on the source cluster, as a user with HBase and HDFS write permission on the destination cluster, and HDFS read permission on the source cluster. This creates the expected amount of IO load on the destination cluster. Optionally, you can limit bandwidth consumption, which affects IO on the destination cluster. After the `ExportSnapshot` operation completes, you can see the snapshot in the new cluster using the `list_snapshot` command, and you can use the `clone_snapshot` command to create the table in the new cluster from the snapshot.

For full instructions for the `snapshot` and `clone_snapshot` HBase Shell commands, run the HBase Shell and type `help snapshot`. The following example takes a snapshot of a table, uses it to clone the table to a new table in the same cluster, and then uses the `ExportSnapshot` utility to copy the table to a different cluster, with 16 mappers and limited to 200 Mb/sec bandwidth.

```

$ bin/hbase shell
hbase(main):005:0> snapshot 'TestTable', 'TestTableSnapshot'
0 row(s) in 2.3290 seconds

hbase(main):006:0> clone_snapshot 'TestTableSnapshot', 'NewTestTable'
0 row(s) in 1.3270 seconds

hbase(main):007:0> describe 'NewTestTable'
DESCRIPTION                               ENABLED
'NewTestTable', {NAME => 'cf1', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}, {NAME => 'cf2', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESSION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
1 row(s) in 0.1280 seconds
hbase(main):008:0> quit

$ hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot TestTableSnapshot -copy-to file:///tmp/hbase -mappers 16 -bandwidth 200
14/10/28 21:48:16 INFO snapshot.ExportSnapshot: Copy Snapshot Manifest
14/10/28 21:48:17 INFO client.RMPProxy: Connecting to ResourceManager at hbase-21.example.com/10.20.188.121:8032
14/10/28 21:48:19 INFO snapshot.ExportSnapshot: Loading Snapshot 'TestTableSnapshot' hfile list
14/10/28 21:48:19 INFO Configuration.deprecation: hadoop.native.lib is deprecated. Instead, use io.native.lib.available
14/10/28 21:48:19 INFO util.FSVisitor: No logs under directory:hdfs://hbase-21.example.com:8020/hbase/.hbase-snapshot/TestTableSnapshot/WALS
14/10/28 21:48:20 INFO mapreduce.JobSubmitter: number of splits:0
14/10/28 21:48:20 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1414556809048_0001
14/10/28 21:48:20 INFO impl.YarnClientImpl: Submitted application application_1414556809048_0001
14/10/28 21:48:20 INFO mapreduce.Job: The url to track the job: http://hbase-21.example.com:8088/proxy/application_1414556809048_0001/
14/10/28 21:48:20 INFO mapreduce.Job: Running job: job_1414556809048_0001
14/10/28 21:48:36 INFO mapreduce.Job: Job job_1414556809048_0001 running in uber mode : false
14/10/28 21:48:36 INFO mapreduce.Job: map 0% reduce 0%
14/10/28 21:48:37 INFO mapreduce.Job: Job job_1414556809048_0001 completed successfully
14/10/28 21:48:37 INFO mapreduce.Job: Counters: 2
Job Counters
  Total time spent by all maps in occupied slots (ms)=0
  Total time spent by all reduces in occupied slots (ms)=0
14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Finalize the Snapshot Export
14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Verify snapshot integrity
14/10/28 21:48:37 INFO Configuration.deprecation: fs.default.name is deprecated. Instead, use fs.defaultFS
14/10/28 21:48:37 INFO snapshot.ExportSnapshot: Export Completed: TestTableSnapshot

```

The bold italic line contains the URL from which you can track the `ExportSnapshot` job. When it finishes, a new set of HFiles, comprising all of the HFiles that were part of the table when the snapshot was taken, is created at the HDFS location you specified.

You can use the `SnapshotInfo` command-line utility included with HBase to verify or debug snapshots.

Using BulkLoad

HBase uses the well-known HFile format to store its data on disk. In many situations, writing HFiles programmatically with your data, and bulk-loading that data into HBase on the RegionServer, has advantages over other data ingest mechanisms. BulkLoad operations bypass the write path completely, providing the following benefits:

- The data is available to HBase immediately but does cause additional load or latency on the cluster when it appears.
- BulkLoad operations do not use the write-ahead log (WAL) and do not cause flushes or split storms.
- BulkLoad operations do not cause excessive garbage collection.



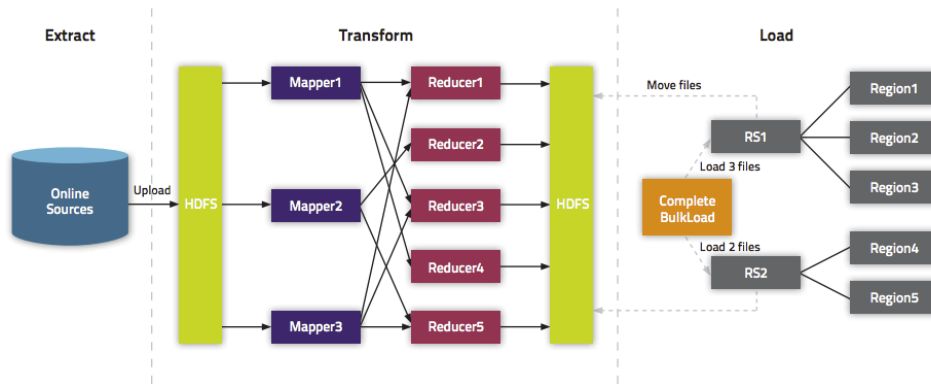
Note: Because they bypass the WAL, BulkLoad operations are not propagated between clusters using replication. If you need the data on all replicated clusters, you must perform the BulkLoad on each cluster.

If you use BulkLoads with HBase, your workflow is similar to the following:

- 1. Extract your data from its existing source.** For instance, if your data is in a MySQL database, you might run the `mysqldump` command. The process you use depends on your data. If your data is already in TSV or CSV format, skip this step and use the included `ImportTsv` utility to process your data into HFiles. See the [ImportTsv documentation](#) for details.
- 2. Process your data into HFile format.** See http://hbase.apache.org/book.html#_hfile_format_2 for details about HFile format. Usually you use a MapReduce job for the conversion, and you often need to write the Mapper yourself because your data is unique. The job must to emit the row key as the `Key`, and either a `KeyValue`, a `Put`, or a `Delete` as the `Value`. The Reducer is handled by HBase; configure it using [HFileOutputFormat.configureIncrementalLoad\(\)](#) and it does the following:
 - Inspects the table to configure a total order partitioner
 - Uploads the partitions file to the cluster and adds it to the `DistributedCache`
 - Sets the number of `reduce` tasks to match the current number of regions
 - Sets the output key/value class to match `HFileOutputFormat` requirements
 - Sets the Reducer to perform the appropriate sorting (either `KeyValueSortReducer` or `PutSortReducer`)
- 3. One HFile is created per region in the output folder.** Input data is almost completely re-written, so you need available disk space at least twice the size of the original data set. For example, for a 100 GB output from `mysqldump`, you should have at least 200 GB of available disk space in HDFS. You can delete the original input file at the end of the process.
- 4. Load the files into HBase.** Use the `LoadIncrementalHFiles` command (more commonly known as the [completebulkload](#) tool), passing it a URL that locates the files in HDFS. Each file is loaded into the relevant region on the RegionServer for the region. You can limit the number of versions that are loaded by passing the `--versions= N` option, where `N` is the maximum number of versions to include, from newest to oldest (largest timestamp to smallest timestamp).

If a region was split after the files were created, the tool automatically splits the HFile according to the new boundaries. This process is inefficient, so if your table is being written to by other processes, you should load as soon as the transform step is done.

The following illustration shows the full BulkLoad process.



Extra Steps for BulkLoad With Encryption Zones

When using BulkLoad to import data into HBase in a cluster using encryption zones, the following information is important.

- Both the staging directory and the directory into which you place your generated HFiles need to be within HBase's encryption zone (generally under the `/hbase` directory). Before you can do this, you need to change the permissions of `/hbase` to be world-executable but not world-readable (`rwx--x--x`, or numeric mode `711`).
- You also need to configure the HMaster to set the permissions of the HBase root directory correctly. If you use Cloudera Manager, edit the **Master Advanced Configuration Snippet (Safety Valve) for `hbase-site.xml`**. Otherwise, edit `hbase-site.xml` on the HMaster. Add the following:

```
<property>
  <name>hbase.rootdir.perms</name>
  <value>711</value>
</property>
```

If you skip this step, a previously-working BulkLoad setup will start to fail with permission errors when you restart the HMaster.

Use Cases for BulkLoad

- **Loading your original dataset into HBase for the first time** - Your initial dataset might be quite large, and bypassing the HBase write path can speed up the process considerably.
- **Incremental Load** - To load new data periodically, use BulkLoad to import it in batches at your preferred intervals. This alleviates latency problems and helps you to achieve service-level agreements (SLAs). However, one trigger for compaction is the number of HFiles on a RegionServer. Therefore, importing a large number of HFiles at frequent intervals can cause major compactions to happen more often than they otherwise would, negatively impacting performance. You can mitigate this by tuning the compaction settings such that the maximum number of HFiles that can be present without triggering a compaction is very high, and relying on other factors, such as the size of the Memstore, to trigger compactions.
- **Data needs to originate elsewhere** - If an existing system is capturing the data you want to have in HBase and needs to remain active for business reasons, you can periodically BulkLoad data from the system into HBase so that you can perform operations on it without impacting the system.

More Information about BulkLoad

For more information and examples, as well as an explanation of the `ImportTsv` utility, which can be used to import data in text-delimited formats such as CSV, see [this post](#) on the Cloudera Blog.

Using Cluster Replication

If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters. In HBase, cluster replication refers to keeping one cluster state synchronized with that of another cluster, using the write-ahead log (WAL) of the source cluster to propagate the changes. Replication is enabled at column family granularity.

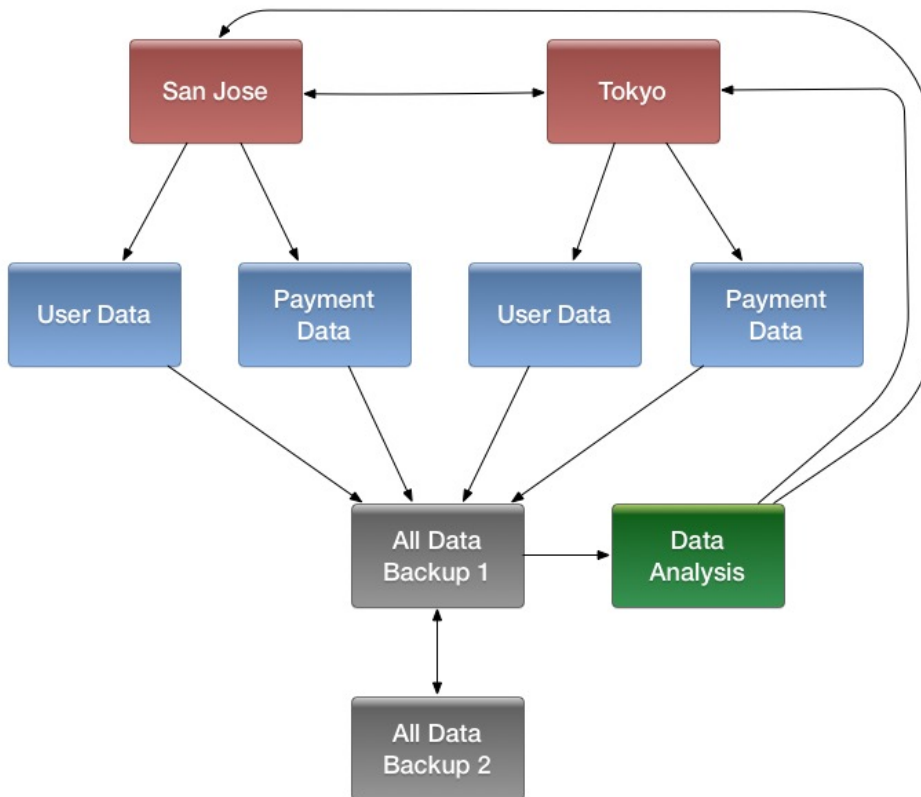
Before enabling replication for a column family, create the table and all column families to be replicated, on the destination cluster.

Cluster replication uses an active-push methodology. An HBase cluster can be a source (also called *active*, meaning that it writes new data), a destination (also called *passive*, meaning that it receives data using replication), or can fulfill both roles at once. Replication is asynchronous, and the goal of replication is consistency.

When data is replicated from one cluster to another, the original source of the data is tracked with a cluster ID, which is part of the metadata. In CDH 5, all clusters that have already consumed the data are also tracked. This prevents replication loops.

Common Replication Topologies

- A central source cluster might propagate changes to multiple destination clusters, for failover or due to geographic distribution.
- A source cluster might push changes to a destination cluster, which might also push its own changes back to the original cluster.
- Many different low-latency clusters might push changes to one centralized cluster for backup or resource-intensive data-analytics jobs. The processed data might then be replicated back to the low-latency clusters.
- Multiple levels of replication can be chained together to suit your needs. The following diagram shows a hypothetical scenario. Use the arrows to follow the data paths.



At the top of the diagram, the `San Jose` and `Tokyo` clusters, shown in red, replicate changes to each other, and each also replicates changes to a `User Data` and a `Payment Data` cluster.

Each cluster in the second row, shown in blue, replicates its changes to the `All Data Backup 1` cluster, shown in grey. The `All Data Backup 1` cluster replicates changes to the `All Data Backup 2` cluster (also shown in grey), as well as the `Data Analysis` cluster (shown in green). `All Data Backup 2` also propagates any of its own changes back to `All Data Backup 1`.

The `Data Analysis` cluster runs MapReduce jobs on its data, and then pushes the processed data back to the `San Jose` and `Tokyo` clusters.

Configuring Clusters for Replication

To configure your clusters for replication, see [HBase Replication](#) on page 380 and [Configuring Secure HBase Replication](#). The following is a high-level overview of the steps to enable replication.



Important: To run replication-related HBase commands, your user must have HBase administrator permissions. If ZooKeeper uses Kerberos, [configure HBase Shell to authenticate to ZooKeeper using Kerberos](#) before attempting to run replication-related commands. No replication-related ACLs are available at this time.

1. Configure and start the source and destination clusters. Create tables with the same names and column families on both the source and destination clusters, so that the destination cluster knows where to store data it receives. All hosts in the source and destination clusters should be reachable to each other.
2. On the source cluster, enable replication in Cloudera Manager, or by setting `hbase.replication` to `true` in `hbase-site.xml`.
3. Obtain Kerberos credentials as the HBase principal. Substitute your `fully.qualified.domain.name` and `realm` in the following command:

```
$ kinit -k -t /etc/hbase/conf/hbase.keytab
hbase/fully.qualified.domain.name@YOUR-REALM.COM
```

4. On the source cluster, in HBase Shell, add the destination cluster as a peer, using the `add_peer` command. The syntax is as follows:

```
add_peer 'ID', 'CLUSTER_KEY'
```

The ID must be a short integer. To compose the `CLUSTER_KEY`, use the following template:

```
hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.znode.parent
```

If both clusters use the same ZooKeeper cluster, you must use a different `zookeeper.znode.parent`, because they cannot write in the same folder.

5. On the source cluster, configure each column family to be replicated by setting its `REPLICATION_SCOPE` to `1`, using commands such as the following in HBase Shell.

```
hbase> disable 'example_table'
hbase> alter 'example_table', {NAME => 'example_family', REPLICATION_SCOPE => '1'}
hbase> enable 'example_table'
```

6. Verify that replication is occurring by examining the logs on the source cluster for messages such as the following.

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 10.10.1.49:62020
```

7. To verify the validity of replicated data, use the included `VerifyReplication` MapReduce job on the source cluster, providing it with the ID of the replication peer and table name to verify. Other options are available, such as a time range or specific families to verify.

The command has the following form:

```
hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication
[--starttime=timestamp] [--stoptime=timestamp] [--families=comma separated list of
families] <peerId> <tablename>
```

The `VerifyReplication` command prints `GOODROWS` and `BADROWS` counters to indicate rows that did and did not replicate correctly.

**Note:**

Some changes are not replicated and must be propagated by other means, such as [Snapshots](#) or [CopyTable](#). See [Initiating Replication When Data Already Exists](#) on page 384 for more details.

- Data that existed in the master before replication was enabled.
- Operations that bypass the WAL, such as when using BulkLoad or API calls such as `writeToWal(false)`.
- Table schema modifications.

Using Pig and HCatalog

Apache Pig is a platform for analyzing large data sets using a high-level language. Apache HCatalog is a sub-project of Apache Hive, which enables reading and writing of data from one Hadoop utility to another. You can use a combination of Pig and HCatalog to import data into HBase. The initial format of your data and other details about your infrastructure determine the steps you follow to accomplish this task. The following simple example assumes that you can get your data into a TSV (text-separated value) format, such as a tab-delimited or comma-delimited text file.

1. Format the data as a TSV file. You can work with other file formats; see the Pig and HCatalog project documentation for more details.

The following example shows a subset of data from [Google's NGram Dataset](#), which shows the frequency of specific phrases or letter-groupings found in publications indexed by Google. Here, the first column has been added to this dataset as the row ID. The first column is formulated by combining the n-gram itself (in this case, `Zones`) with the line number of the file in which it occurs (`z_LINE_NUM`). This creates a format such as "Zones_z_6230867." The second column is the n-gram itself, the third column is the year of occurrence, the fourth column is the frequency of occurrence of that Ngram in that year, and the fifth column is the number of distinct publications. This extract is from the z file of the 1-gram dataset from version 20120701. The data is truncated at the . . . mark, for the sake of readability of this document. In most real-world scenarios, you will not work with tables that have five columns. Most HBase tables have one or two columns.

```
Zones_z_6230867 Zones 1507 1 1
Zones_z_6230868 Zones 1638 1 1
Zones_z_6230869 Zones 1656 2 1
Zones_z_6230870 Zones 1681 8 2
...
Zones_z_6231150 Zones 1996 17868 4356
Zones_z_6231151 Zones 1997 21296 4675
Zones_z_6231152 Zones 1998 20365 4972
Zones_z_6231153 Zones 1999 20288 5021
Zones_z_6231154 Zones 2000 22996 5714
Zones_z_6231155 Zones 2001 20469 5470
Zones_z_6231156 Zones 2002 21338 5946
Zones_z_6231157 Zones 2003 29724 6446
Zones_z_6231158 Zones 2004 23334 6524
Zones_z_6231159 Zones 2005 24300 6580
Zones_z_6231160 Zones 2006 22362 6707
Zones_z_6231161 Zones 2007 22101 6798
Zones_z_6231162 Zones 2008 21037 6328
```

2. Using the `hadoop fs` command, put the data into HDFS. This example places the file into an `/imported_data/` directory.

```
$ hadoop fs -put zones_frequency.tsv /imported_data/
```

3. Create and register a new HBase table in HCatalog, using the `hcat` command, passing it a DDL file to represent your table. You could also register an existing HBase table, using the same command. The DDL file format is specified as part of the [Hive REST API](#). The following example illustrates the basic mechanism.

```
CREATE TABLE
zones_frequency_table (id STRING, ngram STRING, year STRING, freq STRING, sources STRING)
STORED BY 'org.apache.hcatalog.hbase.HBaseHCatStorageHandler'
TBLPROPERTIES (
  'hbase.table.name' = 'zones_frequency_table',
  'hbase.columns.mapping' = 'd:ngram,d:year,d:freq,d:sources',
  'hcat.hbase.output.bulkMode' = 'true'
);
```

```
$ hcat -f zones_frequency_table.ddl
```

4. Create a Pig file to process the TSV file created in step 1, using the DDL file created in step 3. Modify the file names and other parameters in this command to match your values if you use data different from this working example. `USING PigStorage('\t')` indicates that the input file is tab-delimited. For more details about Pig syntax, see the [Pig Latin](#) reference documentation.

```
A = LOAD 'hdfs:///imported_data/zones_frequency.tsv' USING PigStorage('\t') AS
(id:chararray, ngram:chararray, year:chararray, freq:chararray, sources:chararray);
-- DUMP A;
STORE A INTO 'zones_frequency_table' USING org.apache.hcatalog.pig.HCatStorer();
```

Save the file as `zones.bulkload.pig`.

5. Use the `pig` command to bulk-load the data into HBase.

```
$ pig -useHCatalog zones.bulkload.pig
```

The data is now in HBase and is available to use.

Using the Java API

The Java API is the most common mechanism for getting data into HBase, through Put operations. The Thrift and REST APIs, as well as the HBase Shell, use the Java API. The following simple example uses the Java API to put data into an HBase table. The Java API traverses the entire write path and can cause compactions and region splits, which can adversely affect performance.

```
...
HTable table = null;
try {
  table = myCode.createTable(tableName, fam);
  int i = 1;
  List<Put> puts = new ArrayList<Put>();
  for (String labelExp : labelExps) {
    Put put = new Put(Bytes.toBytes("row" + i));
    put.add(fam, qual, HConstants.LATEST_TIMESTAMP, value);
    puts.add(put);
    i++;
  }
  table.put(puts);
} finally {
  if (table != null) {
    table.flushCommits();
  }
}
...
```

Using the Apache Thrift Proxy API

The Apache Thrift library provides cross-language client-server remote procedure calls (RPCs), using *Thrift bindings*. A Thrift binding is client code generated by the Apache Thrift Compiler for a target language (such as Python) that allows communication between the Thrift server and clients using that client code. HBase includes an Apache Thrift Proxy

API, which allows you to write HBase applications in Python, C, C++, or another language that Thrift supports. The Thrift Proxy API is slower than the Java API and may have fewer features. To use the Thrift Proxy API, you need to configure and run the HBase Thrift server on your cluster. See [Installing and Starting the HBase Thrift Server](#). You also need to install the [Apache Thrift compiler](#) on your development system.

After the Thrift server is configured and running, generate Thrift bindings for the language of your choice, using an IDL file. A HBase IDL file named `HBase.thrift` is included as part of HBase. After generating the bindings, copy the Thrift libraries for your language into the same directory as the generated bindings. In the following Python example, these libraries provide the `thrift.transport` and `thrift.protocol` libraries. These commands show how you might generate the Thrift bindings for Python and copy the libraries on a Linux system.

```
$ mkdir HBaseThrift
$ cd HBaseThrift/
$ thrift -gen py /path/to/Hbase.thrift
$ mv gen-py/* .
$ rm -rf gen-py/
$ mkdir thrift
$ cp -rp ~/Downloads/thrift-0.9.0/lib/py/src/* ./thrift/
```

The following example shows a simple Python application using the Thrift Proxy API.

```
from thrift.transport import TSocket
from thrift.protocol import TBinaryProtocol
from thrift.transport import TTransport
from hbase import Hbase

# Connect to HBase Thrift server
transport = TTransport.TBufferedTransport(TSocket.TSocket(host, port))
protocol = TBinaryProtocol.TBinaryProtocolAccelerated(transport)

# Create and open the client connection
client = Hbase.Client(protocol)
transport.open()

# Modify a single row
mutations = [Hbase.Mutation(
    column='columnfamily:columndescriptor', value='columnvalue')]
client.mutateRow('tablename', 'rowkey', mutations)

# Modify a batch of rows
# Create a list of mutations per work of Shakespeare
mutationsbatch = []

for line in myDataFile:
    rowkey = username + "-" + filename + "-" + str(linenum).zfill(6)

    mutations = [
        Hbase.Mutation(column=messagecolumncf, value=line.strip()),
        Hbase.Mutation(column=linenumcolumncf, value=encode(linenum)),
        Hbase.Mutation(column=usernamecolumncf, value=username)
    ]

    mutationsbatch.append(Hbase.BatchMutation(row=rowkey, mutations=mutations))

# Run the mutations for all the lines in myDataFile
client.mutateRows(tablename, mutationsbatch)

transport.close()
```

The Thrift Proxy API does not support writing to HBase clusters that are secured using Kerberos.

This example was modified from the following two blog posts on <http://www.cloudera.com>. See them for more details.

- [Using the HBase Thrift Interface, Part 1](#)
- [Using the HBase Thrift Interface, Part 2](#)

Using the REST Proxy API

After configuring and starting the [HBase REST Server](#) on your cluster, you can use the HBase REST Proxy API to stream data into HBase, from within another application or shell script, or by using an HTTP client such as `wget` or `curl`. The REST Proxy API is slower than the Java API and may have fewer features. This approach is simple and does not require advanced development experience to implement. However, like the Java and Thrift Proxy APIs, it uses the full write path and can cause compactions and region splits.

Specified addresses without existing data create new values. Specified addresses with existing data create new versions, overwriting an existing version if the row, column:qualifier, and timestamp all match that of the existing value.

```
$ curl -H "Content-Type: text/xml" http://localhost:8000/test/testrow/test:testcolumn
```

The REST Proxy API does not support writing to HBase clusters that are secured using Kerberos.

For full documentation and more examples, see the [REST Proxy API documentation](#).

Using Flume

Apache Flume is a fault-tolerant system designed for ingesting data into HDFS, for use with Hadoop. You can configure Flume to write data directly into HBase. Flume includes two different *sinks* designed to work with HBase: `HBaseSink` (`org.apache.flume.sink.hbase.HBaseSink`) and `AsyncHBaseSink` (`org.apache.flume.sink.hbase.AsyncHBaseSink`). `HBaseSink` supports HBase IPC calls introduced in HBase 0.96, and allows you to write data to an HBase cluster that is secured by Kerberos, whereas `AsyncHBaseSink` does not. However, `AsyncHBaseSink` uses an asynchronous model and guarantees atomicity at the row level.

You configure `HBaseSink` and `AsyncHBaseSink` nearly identically. Following is an example configuration for each. Bold lines highlight differences in the configurations. For full documentation about configuring `HBaseSink` and `AsyncHBaseSink`, see the [Flume documentation](#). The `table`, `columnFamily`, and `column` parameters correlate to the HBase table, column family, and column where the data is to be imported. The `serializer` is the class that converts the data at the source into something HBase can use. Configure your sinks in the Flume configuration file.

In practice, you usually need to write your own serializer, which implements either `AsyncHBaseEventSerializer` or `HBaseEventSerializer`. The `HBaseEventSerializer` converts Flume Events into one or more HBase Puts, sends them to the HBase cluster, and is closed when the `HBaseSink` stops. `AsyncHBaseEventSerializer` starts and listens for Events. When it receives an Event, it calls the `setEvent` method and then calls the `getActions` and `getIncrements` methods. When the `AsyncHBaseSink` is stopped, the `serializer` `cleanUp` method is called. These methods return `PutRequest` and `AtomicIncrementRequest`, which are part of the `asynchbase` API.

AsyncHBaseSink:

```
#Use the AsyncHBaseSink
host1.sinks.sink1.type = org.apache.flume.sink.hbase.AsyncHBaseSink
host1.sinks.sink1.channel = chl
host1.sinks.sink1.table = transactions
host1.sinks.sink1.columnFamily = clients
host1.sinks.sink1.column = charges
host1.sinks.sink1.batchSize = 5000
#Use the SimpleAsyncHbaseEventSerializer that comes with Flume
host1.sinks.sink1.serializer = org.apache.flume.sink.hbase.SimpleAsyncHbaseEventSerializer
host1.sinks.sink1.serializer.incrementColumn = icol
host1.channels.chl.type=memory
```

HBaseSink:

```
#Use the HBaseSink
host1.sinks.sink1.type = org.apache.flume.sink.hbase.HBaseSink
host1.sinks.sink1.channel = chl
host1.sinks.sink1.table = transactions
host1.sinks.sink1.columnFamily = clients
host1.sinks.sink1.column = charges
host1.sinks.sink1.batchSize = 5000
#Use the SimpleHbaseEventSerializer that comes with Flume
host1.sinks.sink1.serializer = org.apache.flume.sink.hbase.SimpleHbaseEventSerializer
```

```
host1.sinks.sink1.serializer.incrementColumn = icol
host1.channels.ch1.type=memory
```

The following serializer, taken from an [Apache Flume blog post by Dan Sandler](#), splits the event body based on a delimiter and inserts each split into a different column. The row is defined in the event header. When each event is received, a counter is incremented to track the number of events received.

```
/**
 * A serializer for the AsyncHBaseSink, which splits the event body into
 * multiple columns and inserts them into a row whose key is available in
 * the headers
 */
public class SplittingSerializer implements AsyncHbaseEventSerializer {
    private byte[] table;
    private byte[] colFam;
    private Event currentEvent;
    private byte[][] columnNames;
    private final List<PutRequest> puts = new ArrayList<PutRequest>();
    private final List<AtomicIncrementRequest> incs = new
    ArrayList<AtomicIncrementRequest>();
    private byte[] currentRowKey;
    private final byte[] eventCountCol = "eventCount".getBytes();

    @Override
    public void initialize(byte[] table, byte[] cf) {
        this.table = table;
        this.colFam = cf;
    }

    @Override
    public void setEvent(Event event) {
        // Set the event and verify that the rowKey is not present
        this.currentEvent = event;
        String rowKeyStr = currentEvent.getHeaders().get("rowKey");
        if (rowKeyStr == null) {
            throw new FlumeException("No row key found in headers!");
        }
        currentRowKey = rowKeyStr.getBytes();
    }

    @Override
    public List<PutRequest> getActions() {
        // Split the event body and get the values for the columns
        String eventStr = new String(currentEvent.getBody());
        String[] cols = eventStr.split(",");
        puts.clear();
        for (int i = 0; i < cols.length; i++) {
            //Generate a PutRequest for each column.
            PutRequest req = new PutRequest(table, currentRowKey, colFam,
                columnNames[i], cols[i].getBytes());
            puts.add(req);
        }
        return puts;
    }

    @Override
    public List<AtomicIncrementRequest> getIncrements() {
        incs.clear();
        //Increment the number of events received
        incs.add(new AtomicIncrementRequest(table, "totalEvents".getBytes(), colFam,
            eventCountCol));
        return incs;
    }

    @Override
    public void cleanUp() {
        table = null;
        colFam = null;
        currentEvent = null;
        columnNames = null;
        currentRowKey = null;
    }
}
```

```

}

@Override
public void configure(Context context) {
    //Get the column names from the configuration
    String cols = new String(context.getString("columns"));
    String[] names = cols.split(",");
    byte[][] columnNames = new byte[names.length][];
    int i = 0;
    for(String name : names) {
        columnNames[i++] = name.getBytes();
    }
    this.columnNames = columnNames;
}

@Override
public void configure(ComponentConfiguration conf) {
}
}

```

Using Spark

You can write data to HBase from Apache Spark by using `def saveAsHadoopDataset(conf: JobConf): Unit`. This example is adapted from [a post on the spark-users mailing list](#).

```

// Note: mapred package is used, instead of the
// mapreduce package which contains new hadoop APIs.

import org.apache.hadoop.hbase.mapred.TableOutputFormat
import org.apache.hadoop.hbase.client
// ... some other settings

val conf = HBaseConfiguration.create()

// general hbase settings
conf.set("hbase.rootdir",
        "hdfs://" + nameNodeURL + ":" + hdfsPort + "/hbase")
conf.setBoolean("hbase.cluster.distributed", true)
conf.set("hbase.zookeeper.quorum", hostname)
conf.setInt("hbase.client.scanner.caching", 10000)
// ... some other settings

val jobConfig: JobConf = new JobConf(conf, this.getClass)

// Note: TableOutputFormat is used as deprecated code
// because JobConf is an old hadoop API
jobConfig.setOutputFormat(classOf[TableOutputFormat])
jobConfig.set(TableOutputFormat.OUTPUT_TABLE, outputTable)

```

Next, provide the mapping between how the data looks in Spark and how it should look in HBase. The following example assumes that your HBase table has two column families, `col_1` and `col_2`, and that your data is formatted in sets of three in Spark, like `(row_key, col_1, col_2)`.

```

def convert(triple: (Int, Int, Int)) = {
    val p = new Put(Bytes.toBytes(triple._1))
    p.add(Bytes.toBytes("cf"),
          Bytes.toBytes("col_1"),
          Bytes.toBytes(triple._2))
    p.add(Bytes.toBytes("cf"),
          Bytes.toBytes("col_2"),
          Bytes.toBytes(triple._3))
    (new ImmutableBytesWritable, p)
}

```

To write the data from Spark to HBase, you might use:

```

new PairRDDFunctions(localData.map(convert)).saveAsHadoopDataset(jobConfig)

```

Using Spark and Kafka

This example, written in Scala, uses Apache Spark in conjunction with the Apache Kafka message bus to stream data from Spark to HBase. The example was provided in [SPARK-944](#). It produces some random words and then stores them in an HBase table, creating the table if necessary.

```

package org.apache.spark.streaming.examples

import java.util.Properties

import kafka.producer._

import org.apache.hadoop.hbase.{ HBaseConfiguration, HColumnDescriptor, HTableDescriptor
}
import org.apache.hadoop.hbase.client.{ HBaseAdmin, Put }
import org.apache.hadoop.hbase.io.ImmutableBytesWritable
import org.apache.hadoop.hbase.mapred.TableOutputFormat
import org.apache.hadoop.hbase.mapreduce.TableInputFormat
import org.apache.hadoop.hbase.util.Bytes
import org.apache.hadoop.mapred.JobConf
import org.apache.spark.SparkContext
import org.apache.spark.rdd.{ PairRDDFunctions, RDD }
import org.apache.spark.streaming._
import org.apache.spark.streaming.StreamingContext._
import org.apache.spark.streaming.kafka._

object MetricAggregatorHBase {
  def main(args : Array[String]) {
    if (args.length < 6) {
      System.err.println("Usage: MetricAggregatorTest <master> <zkQuorum> <group> <topics>
<destHBaseTableName> <numThreads>")
      System.exit(1)
    }

    val Array(master, zkQuorum, group, topics, hbaseTableName, numThreads) = args

    val conf = HBaseConfiguration.create()
    conf.set("hbase.zookeeper.quorum", zkQuorum)

    // Initialize hBase table if necessary
    val admin = new HBaseAdmin(conf)
    if (!admin.isTableAvailable(hbaseTableName)) {
      val tableDesc = new HTableDescriptor(hbaseTableName)
      tableDesc.addFamily(new HColumnDescriptor("metric"))
      admin.createTable(tableDesc)
    }

    // setup streaming context
    val ssc = new StreamingContext(master, "MetricAggregatorTest", Seconds(2),
      System.getenv("SPARK_HOME"), StreamingContext.jarOfClass(this.getClass))
    ssc.checkpoint("checkpoint")

    val topicpMap = topics.split(",").map((_, numThreads.toInt)).toMap
    val lines = KafkaUtils.createStream(ssc, zkQuorum, group, topicpMap)
      .map { case (key, value) => ((key, Math.floor(System.currentTimeMillis() /
60000).toLong * 60), value.toInt) }

    val aggr = lines.reduceByKeyAndWindow(add _, Minutes(1), Minutes(1), 2)

    aggr.foreach(line => saveToHBase(line, zkQuorum, hbaseTableName))

    ssc.start

    ssc.awaitTermination
  }

  def add(a : Int, b : Int) = { (a + b) }

  def saveToHBase(rdd : RDD[((String, Long), Int)], zkQuorum : String, tableName :
String) = {
    val conf = HBaseConfiguration.create()
    conf.set("hbase.zookeeper.quorum", zkQuorum)
  }

```



```

val jobConfig = new JobConf(conf)
jobConfig.set(TableOutputFormat.OUTPUT_TABLE, tableName)
jobConfig.setOutputFormat(classOf[TableOutputFormat])

new PairRDDFunctions(rdd.map { case ((metricId, timestamp), value) =>
createHBaseRow(metricId, timestamp, value) }).saveAsHadoopDataset(jobConfig)
}

def createHBaseRow(metricId : String, timestamp : Long, value : Int) = {
  val record = new Put(Bytes.toBytes(metricId + "~" + timestamp))

  record.add(Bytes.toBytes("metric"), Bytes.toBytes("col"),
Bytes.toBytes(value.toString))

  (new ImmutableBytesWritable, record)
}
}

// Produces some random words between 1 and 100.
object MetricDataProducer {

  def main(args : Array[String]) {
    if (args.length < 2) {
      System.err.println("Usage: MetricDataProducer <metadataBrokerList> <topic>
<messagesPerSec>")
      System.exit(1)
    }

    val Array(brokers, topic, messagesPerSec) = args

    // ZooKeeper connection properties
    val props = new Properties()
    props.put("metadata.broker.list", brokers)
    props.put("serializer.class", "kafka.serializer.StringEncoder")

    val config = new ProducerConfig(props)
    val producer = new Producer[String, String](config)

    // Send some messages
    while (true) {
      val messages = (1 to messagesPerSec.toInt).map { messageNum =>
        {
          val metricId = scala.util.Random.nextInt(10)
          val value = scala.util.Random.nextInt(1000)
          new KeyedMessage[String, String](topic, metricId.toString, value.toString)
        }
      }.toArray

      producer.send(messages : _)
      Thread.sleep(100)
    }
  }
}

```

Using a Custom MapReduce Job

Many of the methods to import data into HBase use MapReduce implicitly. If none of those approaches fit your needs, you can use MapReduce directly to convert data to a series of HFiles or API calls for import into HBase. In this way, you can import data from Avro, Parquet, or another format into HBase, or export data from HBase into another format, using API calls such as [TableOutputFormat](#), [HFileOutputFormat](#), and [TableInputFormat](#).

Configuring HBase MultiWAL Support

CDH supports multiple write-ahead logs (MultiWAL) for HBase. (For more information, see [HBASE-5699](#).)

Without MultiWAL support, each region on a RegionServer writes to the same WAL. A busy RegionServer might host several regions, and each write to the WAL is serial because HDFS only supports sequentially written files. This causes the WAL to negatively impact performance.

MultiWAL allows a RegionServer to write multiple WAL streams in parallel by using multiple pipelines in the underlying HDFS instance, which increases total throughput during writes.



Note: In the current implementation of MultiWAL, incoming edits are partitioned by Region. Therefore, throughput to a single Region is not increased.

To configure MultiWAL for a RegionServer, set the value of the property `hbase.wal.provider` to `multiwal` and restart the RegionServer. To disable MultiWAL for a RegionServer, unset the property and restart the RegionServer.

RegionServers using the original WAL implementation and those using the MultiWAL implementation can each handle recovery of either set of WALs, so a zero-downtime configuration update is possible through a rolling restart.

Configuring MultiWAL Support Using Cloudera Manager

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope > RegionServer**.
4. Select **Category > Main**.
5. Set **WAL Provider** to **MultiWAL**.
6. Set the **Per-RegionServer Number of WAL Pipelines** to a value greater than 1.
7. Click **Save Changes** to commit the changes.
8. Restart the RegionServer roles.

Configuring MultiWAL Support Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Edit `hbase-site.xml` on each RegionServer where you want to enable MultiWAL. Add the following property by pasting the XML.

```
<property>
  <name>hbase.wal.provider</name>
  <value>multiwal</value>
</property>
```

2. Stop and restart the RegionServer.

Storing Medium Objects (MOBs) in HBase

Data comes in many sizes, and saving all of your data in HBase, including binary data such as images and documents, is convenient. HBase can technically handle binary objects with cells that are up to 10 MB in size. However, HBase normal read and write paths are optimized for values smaller than 100 KB in size. When HBase handles large numbers of values up to 10 MB (medium objects, or MOBs), performance is degraded because of write amplification caused by splits and compactions.

One way to solve this problem is by storing objects larger than 100KB directly in HDFS, and storing references to their locations in HBase. CDH 5.4 and higher includes optimizations for storing MOBs directly in HBase) based on [HBASE-11339](#).

To use MOB, you must use HFile version 3. Optionally, you can configure the MOB file reader's cache settings Service-Wide and for each RegionServer, and then configure specific columns to hold MOB data. No change to client code is required for HBase MOB support.

Enabling HFile Version 3 Using Cloudera Manager

Minimum Required Role: [Full Administrator](#)

To enable HFile version 3 using Cloudera Manager, edit the HBase Service Advanced Configuration Snippet for HBase Service-Wide.

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Search for the property **HBase Service Advanced Configuration Snippet (Safety Valve)** for `hbase-site.xml`.
4. Paste the following XML into the **Value** field and save your changes.

```
<property>
  <name>hfile.format.version</name>
  <value>3</value>
</property>
```

Changes will take effect after the next major compaction.

Enabling HFile Version 3 Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

Paste the following XML into `hbase-site.xml`.

```
<property>
  <name>hfile.format.version</name>
  <value>3</value>
</property>
```

Restart HBase. Changes will take effect for a given region during its next major compaction.

Configuring Columns to Store MOBs

Use the following options to configure a column to store MOBs:

- `IS_MOB` is a Boolean option, which specifies whether or not the column can store MOBs.
- `MOB_THRESHOLD` configures the number of bytes at which an object is considered to be a MOB. If you do not specify a value for `MOB_THRESHOLD`, the default is 100 KB. If you write a value larger than this threshold, it is treated as a MOB.

You can configure a column to store MOBs using the HBase Shell or the Java API.

Using HBase Shell:

```
hbase> create 't1', {NAME => 'f1', IS_MOB => true, MOB_THRESHOLD => 102400}
hbase> alter 't1', {NAME => 'f1', IS_MOB => true, MOB_THRESHOLD =>
102400}
```

Using the Java API:

```
HColumnDescriptor hcd = new HColumnDescriptor("f");
hcd.setMobEnabled(true);
hcd.setMobThreshold(102400L);
```

HBase MOB Cache Properties

Because there can be a large number of MOB files at any time, as compared to the number of HFiles, MOB files are not always kept open. The MOB file reader cache is a LRU cache which keeps the most recently used MOB files open.

The following properties are available for tuning the HBase MOB cache.

Table 2: HBase MOB Cache Properties

Property	Default	Description
hbase.mob.file.cache.size	1000	The of opened file handlers to cache. A larger value will benefit reads by providing more file handlers per MOB file cache and would reduce frequent file opening and closing of files. However, if the value is too high, errors such as "Too many opened file handlers" may be logged.
hbase.mob.cache.evict.period	3600	The amount of time in seconds after a file is opened before the MOB cache evicts cached files. The default value is 3600 seconds.
hbase.mob.cache.evict.remain.ratio	0.5f	The ratio, expressed as a float between 0.0 and 1.0, that controls how manyfiles remain cached after an eviction is triggered due to the number of cached files exceeding the hbase.mob.file.cache.size. The default value is 0.5f.

Configuring the MOB Cache Using Cloudera Manager

To configure the MOB cache within Cloudera Manager, edit the HBase Service advanced aonfiguration Snippet for the cluster. Cloudera recommends testing your configuration with the default settings first.

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Search for the property **HBase Service Advanced Configuration Snippet (Safety Valve)** for hbase-site.xml.
4. Paste your configuration into the **Value** field and save your changes. The following example sets the hbase.mob.cache.evict.period property to 5000 seconds. See [Table 2: HBase MOB Cache Properties](#) on page 124 for a full list of configurable properties for HBase MOB.

```
<property>
  <name>hbase.mob.cache.evict.period</name>
  <value>5000</value>
</property>
```

5. Restart your cluster for the changes to take effect.

Configuring the MOB Cache Using the Command Line

Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

Because there can be a large number of MOB files at any time, as compared to the number of HFiles, MOB files are not always kept open. The MOB file reader cache is a LRU cache which keeps the most recently used MOB files open.

To customize the configuration of the MOB file reader's cache on each RegionServer, configure the MOB cache properties in the RegionServer's hbase-site.xml. Customize the configuration to suit your environment, and restart or rolling restart the RegionServer. Cloudera recommends testing your configuration with the default settings first. The following

example sets the `hbase.mob.cache.evict.period` property to 5000 seconds. See [Table 2: HBase MOB Cache Properties](#) on page 124 for a full list of configurable properties for HBase MOB.

```
<property>
  <name>hbase.mob.cache.evict.period</name>
  <value>5000</value>
</property>
```

Testing MOB Storage and Retrieval Performance

HBase provides the Java utility `org.apache.hadoop.hbase.IntegrationTestIngestMOB` to assist with testing the MOB feature and deciding on appropriate configuration values for your situation. The utility is run as follows:

```
$ sudo -u hbase hbase org.apache.hadoop.hbase.IntegrationTestIngestMOB \
  -threshold 102400 \
  -minMobDataSize 512 \
  -maxMobDataSize 5120
```

- **threshold** is the threshold at which cells are considered to be MOBs. The default is 1 kB, expressed in bytes.
- **minMobDataSize** is the minimum value for the size of MOB data. The default is 512 B, expressed in bytes.
- **maxMobDataSize** is the maximum value for the size of MOB data. The default is 5 kB, expressed in bytes.

Compacting MOB Files Manually

You can trigger manual compaction of MOB files manually, rather than waiting for them to be triggered by your [configuration](#), using the HBase Shell commands `compact_mob` and `major_compact_mob`. Each of these commands requires the first parameter to be the table name, and takes an optional column family name as the second argument. If the column family is provided, only that column family's files are compacted. Otherwise, all MOB-enabled column families' files are compacted.

```
hbase> compact_mob 't1'
hbase> compact_mob 't1', 'f1'
hbase> major_compact_mob 't1'
hbase> major_compact_mob 't1', 'f1'
```

This functionality is also available using the API, using the `Admin.compact` and `Admin.majorCompact` methods.

Managing HDFS

The section contains configuration tasks for the HDFS service. For information on configuring HDFS for high availability, see [HDFS High Availability](#) on page 286.

Managing Federated Nameservices

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

Cloudera Manager supports the configuration of multiple nameservices managing separate HDFS namespaces, all of which share the storage available on the set of DataNodes. These nameservices are federated, meaning each nameservice is independent and does not require coordination with other nameservices. See [HDFS Federation](#) for more information.

It is simplest to add a second nameservice if high availability is already enabled. The process of enabling high availability creates a nameservice as part of the enable high availability workflow.



Important: Configuring a new nameservice shut downs the services that depend upon HDFS. Once the new nameservice has been started, the services that depend upon HDFS must be restarted, and the client configurations must be redeployed. (This can be done as part of the **Add Nameservice** workflow, as an option.)

Configuring the First Nameservice Using Cloudera Manager

Follow the instructions below to define the first nameservice.

1. Go to the HDFS service.

2. Click the **Configuration** tab.
3. Type `nameservice` in the Search field. The `nameservice` properties for the `NameNode` and `SecondaryNameNode` display.
4. In the **NameNode Nameservice** field, type a name for the `nameservice`. The name must be unique and can contain only alphanumeric characters.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. In the **Mountpoints** field, change the mount point from "/" to a list of mount points that are in the namespace that this `nameservice` will manage. (You can enter this as a comma-separated list — for example, `/hbase`, `/tmp`, `/user` or by clicking the **+** icon to add each mount point in its own field.) You can determine the list of mount points by running the command `hadoop fs -ls /` from the CLI on the `NameNode` host.
6. In the **SecondaryNameNode Nameservice** field, type the `nameservice` name that you provided for the **NameNode Nameservice** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

7. Click **Save Changes** to commit the changes.
8. Click the **Instances** tab. The **Federation and High Availability** section displays with the `nameservice` listed.

Editing the List of Mountpoints for a Nameservice Using Cloudera Manager

1. Go to the HDFS service.
2. Click the **Instances** tab. The **Federation and High Availability** section displays with the `nameservices` listed.
3. Select **Actions** > **Edit**. In the **Mount Points** field, change the mount point to a list of mount points in the namespace that the `nameservice` will manage.
4. Click **OK**.

Adding a Nameservice Using Cloudera Manager

The instructions below for adding a `nameservice` assume that one `nameservice` is already set up. The first `nameservice` can be set up either by [configuring the first nameservice](#) or by enabling [HDFS high availability](#).

1. Go to the HDFS service.
2. Click the **Instances** tab. At the top of this page you should see the **Federation and High Availability** section. If this section does not appear, it means you do not have any `nameservices` configured. You must have one `nameservice` already configured in order to add a `nameservice`.
3. Click the **Add Nameservice** button.
 - a. In the **Nameservice Name** field, enter a name for the new `nameservice`. The name must be unique and can contain only alphanumeric characters.
 - b. In the **Mount Points** field, enter at least one mount point for the `nameservice`. This defines the portion of HDFS that will be managed under the new `nameservice`. (Click the **+** to the right of the field to add a new mount point). You cannot use "/" as a mount point; you must specify HDFS directories by name.
 - The mount points must be unique for this `nameservice`; you cannot specify any of the same mount points you have used for other `nameservices`.
 - You can specify mount points that do not yet exist, and create the corresponding directories in a later step in this procedure.
 - If you want to use a mount point previously associated with another `nameservice` you must first remove that mount point from that service. You can do this using the **Edit** command from the **Actions** menu for that `nameservice`, and later add the mount point to the new `nameservice`.
 - After you have brought up the new `nameservice`, you must create the directories that correspond with the mount points you specified in the new namespace.
 - If a mount point corresponds to a directory that formerly was under a different `nameservice`, you must also move any contents of that directory, if appropriate as described in [step 8](#).

- If an HBase service is set to depend on the federated HDFS service, edit the mount points of the existing nameservice to reference:
 - HBase root directory (default `/hbase`)
 - MapReduce system directory (default `/tmp/mapred/system`)
 - MapReduce JobTracker staging root directory (default value `/user`).
 - c. If you want to configure high availability for the nameservice, leave the **Highly Available** checkbox checked.
 - d. Click **Continue**.
4. Select the hosts on which the new NameNode and Secondary NameNodes will be created. (These must be hosts that are not already running other NameNode or SecondaryNameNode instances, and their `/dfs/mn` and `/dfs/snn` directories should be empty if they exist. Click **Continue**.)
 5. Enter or confirm the directory property values (these will differ depending on whether you are enabling high availability for this nameservice, or not).
 6. Select the **Start Dependent Services** checkbox if you need to create directories or move data onto the new nameservice. Leave this checked if you want the workflow to restart services and redeploy the client configurations as the last steps in the workflow.
 7. Click **Continue**. If the process finished successfully, click **Finish**. The new nameservice displays in the **Federation and High Availability** section in the **Instances** tab of the HDFS service.
 8. Create the directories you want under the new nameservice using the CLI:
 - a. To create a directory in the new namespace, use the command `hadoop fs -mkdir /nameservices/nameservice/directory` where *nameservice* is the new nameservice you just created and *directory* is the directory that corresponds to a mount point you specified.
 - b. To move data from one nameservice to another, use `distcp` or manual `export/import`. `dfs -cp` and `dfs -mv` will not work.
 - c. Verify that the directories and data are where you expect them to be.
 9. Restart the dependent services.



Note: The monitoring configurations at the HDFS level apply to *all* nameservices. If you have two nameservices, it is not possible to disable a check on one but not the other. Likewise, it's not possible to have different event thresholds for the two nameservices.

Also see [Changing a Nameservice Name for Highly Available HDFS Using Cloudera Manager](#) on page 308.

Nameservice and Quorum-based Storage

With Quorum-based Storage, JournalNodes are shared across nameservices. So, if JournalNodes are present in an HDFS service, all nameservices will have Quorum-based Storage enabled. To override this:

- The `dfs.namenode.shared.edits.dir` configuration of the two NameNodes of a high availability nameservice should be configured to include the value of the `dfs.namenode.name.dirs` setting, or
- The `dfs.namenode.edits.dir` configuration of the one NameNode of a non-high availability nameservice should be configured to include the value of the `dfs.namenode.name.dirs` setting.

NameNodes

NameNodes maintain the namespace tree for HDFS and a mapping of file blocks to DataNodes where the data is stored. A simple HDFS cluster can have only one primary NameNode, supported by a secondary NameNode that periodically compresses the NameNode edits log file that contains a list of HDFS metadata modifications. This reduces the amount of disk space consumed by the log file on the NameNode, which also reduces the restart time for the primary NameNode. A [high availability](#) cluster contains two NameNodes: active and standby.

Formatting the NameNode and Creating the `/tmp` Directory

Formatting the NameNode and Creating the `/tmp` Directory Using Cloudera Manager

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

Managing CDH and Managed Services

When you add an HDFS service, the wizard automatically formats the NameNode and creates the `/tmp` directory on HDFS. If you quit the wizard or it does not finish, you can format the NameNode and create the `/tmp` directory outside the wizard by doing these steps:

1. Stop the HDFS service if it is running. See [Starting, Stopping, and Restarting Services](#) on page 39.
2. Click the **Instances** tab.
3. Click the NameNode role instance.
4. Select **Actions > Format**.
5. Start the HDFS service.
6. Select **Actions > Create /tmp Directory**.

Formatting the NameNode and Creating the /tmp Directory Using the Command Line

See [Formatting the NameNode](#).

Backing Up and Restoring HDFS Metadata

Backing Up HDFS Metadata Using Cloudera Manager

HDFS metadata backups can be used to restore a NameNode when both NameNode roles have failed. In addition, Cloudera recommends backing up HDFS metadata before a major upgrade.

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

This backup method requires you to shut down the cluster.

1. Note the active NameNode.
2. Stop the cluster. It is particularly important that the NameNode role process is not running so that you can make a consistent backup.
3. Go to the HDFS service.
4. Click the **Configuration** tab.
5. In the Search field, search for "NameNode Data Directories" and note the value.
6. On the active NameNode host, back up the directory listed in the NameNode Data Directories property. If more than one is listed, make a backup of one directory, since each directory is a complete copy. For example, if the NameNode data directory is `/data/dfs/nn`, do the following as root:

```
# cd /data/dfs/nn
# tar -cvf /root/nn_backup_data.tar .
```

You should see output like this:

```
./
./current/
./current/fsimage
./current/fstime
./current/VERSION
./current/edits
./image/
./image/fsimage
```

If there is a file with the extension `lock` in the NameNode data directory, the NameNode most likely is still running. Repeat the steps, starting by shutting down the NameNode role.

Restoring HDFS Metadata From a Backup Using Cloudera Manager

The following process assumes a scenario where both NameNode hosts have failed and you must restore from a backup.

1. Remove the NameNode, JournalNode, and Failover Controller roles from the HDFS service.
2. Add the host on which the NameNode role will run.
3. Create the NameNode data directory, ensuring that the permissions, ownership, and group are set correctly.
4. Copy the backed up files to the NameNode data directory.
5. Add the NameNode role to the host.

6. Add the Secondary NameNode role to another host.
7. Enable high availability. If not all roles are started after the wizard completes, restart the HDFS service. Upon startup, the NameNode reads the fsimage file and loads it into memory. If the JournalNodes are up and running and there are edit files present, any edits newer than the fsimage are applied.

Moving NameNode Roles

This section describes two procedures for moving NameNode roles. Both procedures require cluster downtime. If [highly availability](#) is enabled for the NameNode, you can use a Cloudera Manager wizard to automate the migration process. Otherwise you must manually delete and add the NameNode role to a new host.

After moving a NameNode, if you have a Hive or Impala service, perform the steps in [NameNode Post-Migration Steps](#) on page 130.

Moving Highly Available NameNode, Failover Controller, and JournalNode Roles Using the Migrate Roles Wizard

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

The Migrate Roles wizard allows you to move roles of a highly available HDFS service from one host to another. You can use it to move NameNode, JournalNode, and Failover Controller roles.

Requirements and Limitations

- Nameservice [federation](#) (multiple namespaces) is *not supported*.
- This procedure requires cluster downtime, not a shutdown. The services discussed in this list must be running for the migration to complete.
- The configuration of HDFS and services that depend on it must be valid.
- The destination host must be commissioned and healthy.
- The NameNode must be highly available using quorum-based storage.
- HDFS automatic failover must be enabled, and the cluster must have a running ZooKeeper service.
- If a Hue service is present in the cluster, its HDFS Web Interface Role property must refer to an HttpFS role, not to a NameNode role.
- A majority of configured JournalNode roles must be running.
- The Failover Controller role that is not located on the source host must be running.

Before You Begin

Do the following before you run the wizard:

- On hosts running active and standby NameNodes, back up the data directories.
- On hosts running JournalNodes, back up the JournalNode edits directory.
- If the source host is not functioning properly, or is not reliably reachable, decommission the host.
- If CDH and HDFS metadata was recently upgraded, and the metadata upgrade was not finalized, finalize the metadata upgrade.

Running the Migrate Roles Wizard

1. If the host to which you want to move the NameNode is not in the cluster, follow the instructions in [Adding a Host to the Cluster](#) on page 54 to add the host.
2. Go to the HDFS service.
3. Click the **Instances** tab.
4. Click the **Migrate Roles** button.
5. Click the **Source Host** text field and specify the host running the roles to migrate. In the Search field optionally enter hostnames to filter the list of hosts and click **Search**.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com


- IP addresses
- Rack name

Select the checkboxes next to the desired host. The list of available roles to migrate displays. Deselect any roles you do not want to migrate. When migrating a NameNode, the co-located Failover Controller must be migrated as well.

6. Click the **Destination Host** text field and specify the host to which the roles will be migrated. On destination hosts, indicate whether to delete data in the NameNode data directories and JournalNode edits directory. If you choose not to delete data and such role data exists, the Migrate Roles command will not complete successfully.
7. Acknowledge that the migration process incurs service unavailability by selecting the **Yes, I am ready to restart the cluster now** checkbox.
8. Click **Continue**. The Command Progress screen displays listing each step in the migration process.
9. When the migration completes, click **Finish**.

Moving a NameNode to a Different Host Using Cloudera Manager

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



Note: This procedure requires cluster downtime.

1. If the host to which you want to move the NameNode is not in the cluster, follow the instructions in [Adding a Host to the Cluster](#) on page 54 to add the host.
2. [Stop all cluster services](#).
3. Make a backup of the `dfs.name.dir` directories on the existing NameNode host. Make sure you back up the `fsimage` and `edits` files. They should be the same across all of the directories specified by the `dfs.name.dir` property.
4. Copy the files you backed up from `dfs.name.dir` directories on the old NameNode host to the host where you want to run the NameNode.
5. Go to the HDFS service.
6. Click the **Instances** tab.
7. Select the checkbox next to the NameNode role instance and then click the **Delete** button. Click **Delete** again to confirm.
8. In the **Review configuration changes** page that appears, click **Skip**.
9. Click **Add Role Instances** to add a NameNode role instance.
10. Select the host where you want to run the NameNode and then click **Continue**.
11. Specify the location of the `dfs.name.dir` directories where you copied the data on the new host, and then click **Accept Changes**.
12. [Start cluster services](#). After the HDFS service has started, Cloudera Manager distributes the new configuration files to the DataNodes, which will be configured with the IP address of the new NameNode host.

NameNode Post-Migration Steps

After moving a NameNode, if you have a Hive or Impala service, perform the following steps:

1. Go to the Hive service.
2. Stop the Hive service.
3. Select **Actions > Update Hive Metastore NameNodes**.

4. If you have an Impala service, restart the Impala service or run an [INVALIDATE METADATA](#) query.

DataNodes

DataNodes store data in a Hadoop cluster and is the name of the daemon that manages the data. File data is replicated on multiple DataNodes for reliability and so that localized computation can be executed near the data.

How NameNode Manages Blocks on a Failed DataNode

After a period without any heartbeats (which by default is 10.5 minutes), a DataNode is assumed to be failed. The following describes how the NameNode manages block replication in such cases.

1. NameNode determines which blocks were on the failed DataNode.
2. NameNode locates other DataNodes with copies of these blocks.
3. The DataNodes with block copies are instructed to copy those blocks to other DataNodes to maintain the configured replication factor.
4. Follow the procedure in [Replacing a Disk on a DataNode Host](#) on page 131 or [Performing Disk Hot Swap for DataNodes](#) on page 133 to bring a repaired DataNode back online.

Replacing a Disk on a DataNode Host

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

For CDH 5.3 and higher, see [Performing Disk Hot Swap for DataNodes](#) on page 133.

If one of your DataNode hosts experiences a disk failure, follow this process to replace the disk:

1. Stop managed services.
2. [Decommission](#) the DataNode role instance.
3. Replace the failed disk.
4. Recommission the DataNode role instance.
5. Run the HDFS `fsck` utility to validate the health of HDFS. The utility normally reports over-replicated blocks immediately after a DataNode is reintroduced to the cluster, which is automatically corrected over time.
6. Start managed services.

Adding and Removing Storage Directories for DataNodes

Adding and Removing Storage Directories Using Cloudera Manager

Adding Storage Directories

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

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope** > **DataNode**.
4. Add the new storage directory to the **DataNode Data Directory** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Removing Storage Directories

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

1. Stop the cluster.
2. Go to the HDFS service.
3. Click the **Configuration** tab.
4. Select **Scope** > **DataNode**.
5. Remove the current directories and add new ones to the **DataNode Data Directory** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Copy the contents under the old directory to the new directory.
8. Start the cluster.

Configuring Storage-Balancing for DataNodes

You can configure HDFS to distribute writes on each DataNode in a manner that balances out available storage among that DataNode's disk volumes.

By default a DataNode writes new block replicas to disk volumes solely on a round-robin basis. You can configure a volume-choosing policy that causes the DataNode to take into account how much space is available on each volume when deciding where to place a new replica.

You can configure

- how much DataNode volumes are allowed to differ in terms of bytes of free disk space before they are considered imbalanced, *and*
- what percentage of new block allocations will be sent to volumes with more available disk space than others.

Configuring Storage-Balancing for DataNodes Using Cloudera Manager

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

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > DataNode**.
4. Select **Category > Advanced**.
5. Configure the following properties (you can use the Search box to locate the properties):

Property	Value	Description
dfs.datanode.fsdataset.volume.choosing.policy	org.apache.hadoop.hdfs.server.datanode.fsdataset.AvailableSpaceVolumeChoosingPolicy	Enables storage balancing among the DataNode's volumes.
dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold	10737418240 (default)	The amount by which volumes are allowed to differ from each other in terms of bytes of free disk space before they are considered imbalanced. The default is 10737418240 (10 GB). If the free space on each volume is within this range of the other volumes, the volumes will be considered balanced and block assignments will be done on a pure round-robin basis.
dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction	0.75 (default)	What proportion of new block allocations will be sent to volumes with more available disk space than others. The allowable range is 0.0-1.0, but set it in the range 0.5 - 1.0 (that is, 50-100%), since there should be no reason to prefer that volumes with less available disk space receive more block allocations.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Restart the role.

Configuring Storage-Balancing for DataNodes Using the Command Line

See [Configuring Storage-Balancing for DataNodes](#).

Performing Disk Hot Swap for DataNodes

This section describes how to replace HDFS disks without shutting down a DataNode. This is referred to as **hot swap**.

**Warning: Requirements and Limitations**

- Hot swap is supported for CDH 5.4 and higher.
- Hot swap can only add disks with empty data directories.
- Removing a disk does not move the data off the disk, which could potentially result in data loss.
- Do not perform hot swap on multiple hosts at the same time.

Performing Disk Hot Swap for DataNodes Using Cloudera Manager

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

1. Configure data directories to remove the disk you are swapping out:
 - a. Go to the HDFS service.
 - b. Click the **Instances** tab.
 - c. In the **Role Type** column, click on the affected DataNode.
 - d. Click the **Configuration** tab.
 - e. Select **Scope > DataNode**.
 - f. Select **Category > Main**.
 - g. Change the value of the **DataNode Data Directory** property to remove the directories that are mount points for the disk you are removing.



Warning: Change the value of this property only for the specific DataNode instance where you are planning to hot swap the disk. *Do not* edit the role group value for this property. Doing so will cause data loss.

2. Click **Save Changes** to commit the changes.
3. Refresh the affected DataNode. Select **Actions > Refresh Data Directories**.
4. Remove the old disk and add the replacement disk.
5. Change the value of the **DataNode Data Directory** property to add back the directories that are mount points for the disk you added.
6. Click **Save Changes** to commit the changes.
7. Refresh the affected DataNode. Select **Actions > Refresh Data Directories**.
8. Run the HDFS `fsck` utility to validate the health of HDFS.

Performing Disk Hot Swap for DataNodes Using the Command Line**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

Use these instructions to perform hot swap of disks in a cluster that is not managed by Cloudera Manager

To add and remove disks:

1. If you are adding disks, format and mount them.

2. Change the value of `dfs.datanode.data.dir` in `hdfs-site.xml` on the DataNode to reflect the directories that will be used from now on (add new points and remove obsolete ones). For more information, see the instructions for DataNodes under [Configuring Local Storage Directories](#).
3. Start the reconfiguration process:

- If Kerberos is enabled:

```
$ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM> &&
dfsadmin -reconfig datanode HOST:PORT start
```

- If Kerberos is not enabled:

```
$ sudo -u hdfs hdfs dfsadmin -reconfig datanode HOST:PORT start
```

where `HOST:PORT` is the DataNode's `dfs.datanode.ipc.address` (or its hostname and the port specified in `dfs.datanode.ipc.address`; for example `dnhost1.example.com:5678`)

To check on the progress of the reconfiguration, you can use the `status` option of the command; for example, if Kerberos is not enabled:

```
$ sudo -u hdfs hdfs dfsadmin -reconfig datanode HOST:PORT status
```

4. Once the reconfiguration is complete, unmount any disks you have removed from the configuration.
5. Run the HDFS `fsck` utility to validate the health of HDFS.

To perform maintenance on a disk:

1. Change the value of `dfs.datanode.data.dir` in `hdfs-site.xml` on the DataNode to exclude the mount point directories that reside on the affected disk and reflect only the directories that will be used during the maintenance window. For more information, see the instructions for DataNodes under [Configuring Local Storage Directories](#).
2. Start the reconfiguration process:

- If Kerberos is enabled:

```
$ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM> &&
dfsadmin -reconfig datanode HOST:PORT start
```

- If Kerberos is not enabled:

```
$ sudo -u hdfs hdfs dfsadmin -reconfig datanode HOST:PORT start
```

where `HOST:PORT` is the DataNode's `dfs.datanode.ipc.address`, or its hostname and the port specified in `dfs.datanode.ipc.address`.

To check on the progress of the reconfiguration, you can use the `status` option of the command; for example, if Kerberos is not enabled:

```
$ sudo -u hdfs hdfs dfsadmin -reconfig datanode HOST:PORT status
```

3. Once the reconfiguration is complete, unmount the disk.
4. Perform maintenance on the disk.
5. Remount the disk.
6. Change the value of `dfs.datanode.data.dir` again to reflect the original set of mount points.
7. Repeat step 2.
8. Run the HDFS `fsck` utility to validate the health of HDFS.

JournalNodes

High-availability clusters use JournalNodes to synchronize active and standby NameNodes. The active NameNode writes to each JournalNode with changes, or "edits," to HDFS namespace metadata. During failover, the standby NameNode applies all edits from the JournalNodes before promoting itself to the active state.

Moving the JournalNode Edits Directory

Moving the JournalNode Edits Directory for an Role Instance Using Cloudera Manager

To change the location of the edits directory for one JournalNode instance:

1. Reconfigure the **JournalNode Edits Directory**.
 - a. Go to the **HDFS** service in Cloudera Manager.
 - b. Click **JournalNode** under **Status Summary**.
 - c. Click the **JournalNode** link for the instance you are changing.
 - d. Click the **Configuration** tab.
 - e. Set `dfs.journalnode.edits.dir` to the path of the new `jn` directory.
 - f. Click Save Changes.
2. Move the location of the JournalNode (`jn`) directory at the command line:
 - a. Connect to host of the JournalNode.
 - b. Copy the JournalNode (`jn`) directory to its new location with the `-a` option to preserve permissions:

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

- c. Rename the old `jn` directory to avoid confusion:

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

3. Redeploy the HDFS client configuration:
 - a. Go to the **HDFS** service.
 - b. Select **Actions** > **Deploy Client Configuration**.
4. Perform a [Rolling Restart](#) on page 41 for HDFS by selecting **Actions** > **Rolling Restart**. Use the default settings.
5. From the command line, delete the old `jn_to_delete` directory.

Moving the JournalNode Edits Directory for a Role Group Using Cloudera Manager

To change the location of the edits directory for each JournalNode in the JournalNode Default Group:

1. Stop all services on the cluster in Cloudera Manager:
 - a. Go to the **Cluster**.
 - b. Select **Actions** > **Stop**.
2. Find the list of JournalNode hosts:
 - a. Go to the **HDFS** service.
 - b. Click **JournalNode** under **Status Summary**.
3. Move the location of each JournalNode (`jn`) directory at the command line:
 - a. Connect to each host with a JournalNode.
 - b. Per host, copy the JournalNode (`jn`) directory to its new location with the `-a` option to preserve permissions:

```
cp -a /<old_path_to_jn_dir>/jn /<new_path_to_jn_dir>/jn
```

- c. Per host, rename the old `jn` directory to avoid confusion:

```
mv /<old_path_to_jn_dir>/jn /<old_path_to_jn_dir>/jn_to_delete
```

4. Reconfigure the **JournalNode Default Group**:

- a. Go to the **HDFS** service.
- b. Click the **Configuration** tab.
- c. Click **JournalNode** under **Scope**.
- d. Set `dfs.journalnode.edits.dir` to the path of the new `jn` directory for all JournalNodes in the group.
- e. Click **Save Changes**.

5. Redeploy the client configuration for the cluster:

- a. Go to the **Cluster**.
- b. Select **Actions > Deploy Client Configuration**.

6. Start all services on the cluster by selecting **Actions > Start**.

7. Delete the old `jn_to_delete` directories from the command line.

Moving JournalNodes Across Hosts

To move JournalNodes to a new host, see [Moving Highly Available NameNode, Failover Controller, and JournalNode Roles Using the Migrate Roles Wizard](#) on page 129.

Configuring Short-Circuit Reads

So-called "short-circuit" reads bypass the DataNode, allowing a client to read the file directly, as long as the client is co-located with the data. Short-circuit reads provide a substantial performance boost to many applications and help improve HBase random read profile and Impala performance.

Short-circuit reads require `libhadoop.so` (the [Hadoop Native Library](#)) to be accessible to both the server and the client. `libhadoop.so` is not available if you have installed from a tarball. You must install from an `.rpm`, `.deb`, or `parcel` in order to use short-circuit local reads.

Configuring Short-Circuit Reads Using Cloudera Manager

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



Note: Short-circuit reads are enabled by default in Cloudera Manager.

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > Gateway or HDFS (Service-Wide)**.
4. Select **Category > Performance**.
5. Locate the **Enable HDFS Short Circuit Read** property or search for it by typing its name in the Search box. Check the box to enable it.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Configuring Short-Circuit Reads Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

Configure the following properties in `hdfs-site.xml` to enable short-circuit reads in a cluster that is not managed by Cloudera Manager:

```
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>true</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.size</name>
  <value>1000</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.expiry.ms</name>
  <value>10000</value>
</property>

<property>
  <name>dfs.domain.socket.path</name>
  <value>/var/run/hadoop-hdfs/dn._PORT</value>
</property>
```



Note: The text `_PORT` appears just as shown; you do not need to substitute a number.

If `/var/run/hadoop-hdfs/` is group-writable, make sure its group is `root`.

Configuring HDFS Trash

The Hadoop trash feature helps prevent accidental deletion of files and directories. If trash is enabled and a file or directory is deleted using the Hadoop shell, the file is moved to the `.Trash` directory in the user's home directory instead of being deleted. Deleted files are initially moved to the `Current` sub-directory of the `.Trash` directory, and their original path is preserved. If trash checkpointing is enabled, the `Current` directory is periodically renamed using a timestamp. Files in `.Trash` are permanently removed after a user-configurable time delay. Files and directories in the trash can be restored simply by moving them to a location outside the `.Trash` directory.

**Important:**

- The trash feature is disabled by default. Cloudera recommends that you enable it on all production clusters.
- The trash feature works by default only for files and directories deleted using the Hadoop shell. Files or directories deleted programmatically using other interfaces (WebHDFS or the Java APIs, for example) are not moved to trash, even if trash is enabled, unless the program has implemented a call to the trash functionality. (Hue, for example, implements trash as of CDH 4.4.)

Users can bypass trash when deleting files using the shell by specifying the `-skipTrash` option to the `hadoop fs -rm -r` command. This can be useful when it is necessary to delete files that are too large for the user's quota.

Configuring HDFS Trash Using Cloudera Manager

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

Enabling and Disabling Trash

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > Gateway**.
4. Select or deselect the **Use Trash** checkbox.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the cluster and deploy the cluster client configuration.

Setting the Trash Interval

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > NameNode**.
4. Specify the **Filesystem Trash Interval** property, which controls the number of minutes after which a trash checkpoint directory is deleted and the number of minutes between trash checkpoints. For example, to enable trash so that deleted files are deleted after 24 hours, set the value of the **Filesystem Trash Interval** property to 1440.



Note: The trash interval is measured from the point at which the files are moved to trash, not from the last time the files were modified.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart all NameNodes.

Configuring HDFS Trash Using the Command Line

See [Enabling Trash](#).

HDFS Balancers

HDFS data might not always be distributed uniformly across DataNodes. One common reason is addition of new DataNodes to an existing cluster. HDFS provides a balancer utility that analyzes block placement and balances data across the DataNodes. The balancer moves blocks until the cluster is deemed to be balanced, which means that the utilization of every DataNode (ratio of used space on the node to total capacity of the node) differs from the utilization of the cluster (ratio of used space on the cluster to total capacity of the cluster) by no more than a given threshold percentage. The balancer does not balance between individual volumes on a single DataNode.

Configuring and Running the HDFS Balancer Using Cloudera Manager

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

In Cloudera Manager, the HDFS balancer utility is implemented by the Balancer role. The Balancer role usually shows a health of **None** on the HDFS Instances tab because it does not run continuously.

The Balancer role is normally added (by default) when the HDFS service is installed. If it has not been added, you must add a Balancer role in order to rebalance HDFS and to see the **Rebalance** action.

Configuring the Balancer Threshold

The Balancer has a default threshold of 10%, which ensures that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%. For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that DataNode disk usage is between 30% and 50% of the DataNode disk-storage capacity. To change the threshold:

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > Balancer**.
4. Select **Category > Main**.
5. Set the **Rebalancing Threshold** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Running the Balancer

1. Go to the HDFS service.
2. Ensure the service has a Balancer role.
3. Select **Actions > Rebalance**.
4. Click **Rebalance** to confirm. If you see a **Finished** status, the Balancer ran successfully.

Configuring and Running the HDFS Balancer Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

The HDFS balancer re-balances data across the DataNodes, moving blocks from overutilized to underutilized nodes. As the system administrator, you can run the balancer from the command-line as necessary -- for example, after adding new DataNodes to the cluster.

Points to note:

- The balancer requires the capabilities of an HDFS superuser (for example, the `hdfs` user) to run.
- The balancer does not balance between individual volumes on a single DataNode.
- You can run the balancer without parameters, as follows:

```
sudo -u hdfs hdfs balancer
```



Note: If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> hadoop <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

This runs the balancer with a default threshold of 10%, meaning that the script will ensure that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%. For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that each DataNode's disk usage is between 30% and 50% of that DataNode's disk-storage capacity.

- You can run the script with a different threshold; for example:

```
sudo -u hdfs hdfs balancer -threshold 5
```

This specifies that each DataNode's disk usage must be (or will be adjusted to be) within 5% of the cluster's overall usage.

- You can adjust the network bandwidth used by the balancer, by running the `dfsadmin -setBalancerBandwidth` command before you run the balancer; for example:

```
dfsadmin -setBalancerBandwidth newbandwidth
```

where *newbandwidth* is the maximum amount of network bandwidth, in bytes per second, that each DataNode can use during the balancing operation. For more information about the bandwidth command, see [BalancerBandwidthCommand](#).

- The balancer can take a long time to run, especially if you are running it for the first time, or do not run it regularly.

Enabling WebHDFS

Enabling WebHDFS Using Cloudera Manager

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

To enable WebHDFS, proceed as follows:

1. Select the HDFS service.
2. Click the **Configuration** tab.
3. Select **Scope > HDFS-1 (Service Wide)**
4. Select the **Enable WebHDFS** property.
5. Click the **Save Changes** button.
6. Restart the HDFS service.

WebHDFS uses the following prefix and URI format: `webhdfs://<HOST>:<HTTP_PORT>/<PATH>`

Secure WebHDFS uses the following prefix and URI format: `webhdfs://<HOST>:<HTTP_PORT>/<PATH>`

You can find a full explanation of the WebHDFS API in the [WebHDFS API documentation](#).

Enabling WebHDFS Using the Command Line

See [Enabling WebHDFS](#).

Adding HttpFS

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

Apache Hadoop HttpFS is a service that provides HTTP access to HDFS.

HttpFS has a REST HTTP API supporting all HDFS filesystem operations (both read and write).

Common HttpFS use cases are:

- Read and write data in HDFS using HTTP utilities (such as `curl` or `wget`) and HTTP libraries from languages other than Java (such as Perl).
- Transfer data between HDFS clusters running different versions of Hadoop (overcoming RPC versioning issues), for example using Hadoop DistCp.
- Read and write data in HDFS in a cluster behind a firewall. (The HttpFS server acts as a gateway and is the only system that is allowed to send and receive data through the firewall).

HttpFS supports Hadoop pseudo-authentication, HTTP SPNEGO Kerberos, and additional authentication mechanisms using a plugin API. HttpFS also supports Hadoop proxy user functionality.

The `webhdfs` client file system implementation can access HttpFS using the Hadoop filesystem command (`hadoop fs`), by using Hadoop DistCp, and from Java applications using the Hadoop file system Java API.

The HttpFS HTTP REST API is interoperable with the WebHDFS REST HTTP API.

For more information about HttpFS, see [Hadoop HDFS over HTTP](#).

The HttpFS role is required for Hue when you enable [HDFS high availability](#).

Adding the HttpFS Role

1. Go to the HDFS service.
2. Click the **Instances** tab.
3. Click **Add Role Instances**.
4. Click the text box below the **HttpFS** field. The Select Hosts dialog box displays.
5. Select the host on which to run the role and click **OK**.
6. Click **Continue**.
7. Check the checkbox next to the **HttpFS** role and select **Actions for Selected > Start**.

Using Load Balancer with HttpFS

To configure a load balancer, select **Clusters > HDFS > Configuration > Category > HttpFS** and enter the hostname and port number of the load balancer in the **HTTPFS Load Balancer** property in the format `hostname:port number`.



Note:

When you set this property, Cloudera Manager regenerates the keytabs for HttpFS roles. The principal in these keytabs contains the load balancer hostname.

If there is a Hue service that depends on this HDFS service, the Hue service has the option to use the load balancer as its HDFS Web Interface Role.

Adding and Configuring an NFS Gateway

The NFSv3 gateway allows a client to mount HDFS as part of the client's local file system. The gateway machine can be any host in the cluster, including the NameNode, a DataNode, or any HDFS client. The client can be any NFSv3-client-compatible machine.



Important:

HDFS does not currently provide ACL support for an NFS gateway.

After mounting HDFS to his or her local filesystem, a user can:

- Browse the HDFS file system as though it were part of the local file system
- Upload and download files from the HDFS file system to and from the local file system.
- Stream data directly to HDFS through the mount point.

File append is supported, but random write is not.

Adding and Configuring an NFS Gateway Using Cloudera Manager

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

The NFS Gateway role implements an NFSv3 gateway. It is an optional role for a CDH 5 HDFS service.

Requirements and Limitations

- The NFS gateway works only with the following operating systems and Cloudera Manager and CDH versions:
 - With Cloudera Manager 5.0.1 or later and CDH 5.0.1 or later, the NFS gateway works on all operating systems supported by Cloudera Manager.
 - With Cloudera Manager 5.0.0 or CDH 5.0.0, the NFS gateway only works on RHEL and similar systems.
 - The NFS gateway is not supported on versions earlier than Cloudera Manager 5.0.0 and CDH 5.0.0.
- If any NFS server is already running on the NFS Gateway host, it must be stopped before the NFS Gateway role is started.

- There are two configuration options related to NFS Gateway role: **Temporary Dump Directory** and **Allowed Hosts and Privileges**. The **Temporary Dump Directory** is automatically created by the NFS Gateway role and should be configured before starting the role.
- The **Access Time Precision** property in the HDFS service must be enabled.

Adding and Configuring the NFS Gateway Role

1. Go to the HDFS service.
2. Click the **Instances** tab.
3. Click **Add Role Instances**.
4. Click the text box below the **NFS Gateway** field. The Select Hosts dialog box displays.
5. Select the host on which to run the role and click **OK**.
6. Click **Continue**.
7. Click the **NFS Gateway** role.
8. Click the **Configuration** tab.
9. Select **Scope > NFS Gateway**.
10. Select **Category > Main**.
11. Ensure that the requirements on the directory set in the **Temporary Dump Directory** property are met.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

12. Optionally edit **Allowed Hosts and Privileges**.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

13. Click **Save Changes** to commit the changes.
14. Click the **Instances** tab.
15. Check the checkbox next to the **NFS Gateway** role and select **Actions for Selected > Start**.

Configuring an NFSv3 Gateway Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

The subsections that follow provide information on installing and configuring the gateway.



Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper`/`YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see [Installing the Latest CDH 5 Release](#) and [Upgrading Unmanaged CDH Using the Command Line](#).

Upgrading from a CDH 5 Beta Release

If you are upgrading from a CDH 5 Beta release, you must first remove the `hadoop-hdfs-portmap` package. Proceed as follows.

1. Unmount existing HDFS gateway mounts. For example, on each client, assuming the file system is mounted on `/hdfs_nfs_mount`:

```
$ umount /hdfs_nfs_mount
```

2. Stop the services:

```
$ sudo service hadoop-hdfs-nfs3 stop
$ sudo hadoop-hdfs-portmap stop
```

3. Remove the `hadoop-hdfs-portmap` package.

- On a RHEL-compatible system:

```
$ sudo yum remove hadoop-hdfs-portmap
```

- On a SLES system:

```
$ sudo zypper remove hadoop-hdfs-portmap
```

- On an Ubuntu or Debian system:

```
$ sudo apt-get remove hadoop-hdfs-portmap
```

4. Install the new version

- On a RHEL-compatible system:

```
$ sudo yum install hadoop-hdfs-nfs3
```

- On a SLES system:

```
$ sudo zypper install hadoop-hdfs-nfs3
```

- On an Ubuntu or Debian system:

```
$ sudo apt-get install hadoop-hdfs-nfs3
```

5. Start the system default portmapper service:

```
$ sudo service portmap start
```

6. Now proceed with [Starting the NFSv3 Gateway](#) on page 145, and then [remount the HDFS gateway mounts](#).*Installing the Packages for the First Time***On RHEL and similar systems:**

Install the following packages on the cluster host you choose for NFSv3 Gateway machine (we'll refer to it as the NFS server from here on).

- `nfs-utils`
- `nfs-utils-lib`
- `hadoop-hdfs-nfs3`

The first two items are standard NFS utilities; the last is a CDH package.

Use the following command:

```
$ sudo yum install nfs-utils nfs-utils-lib hadoop-hdfs-nfs3
```

On SLES:

Install `nfs-utils` on the cluster host you choose for NFSv3 Gateway machine (referred to as the NFS server from here on):

```
$ sudo zypper install nfs-utils
```

On an Ubuntu or Debian system:

Install `nfs-common` on the cluster host you choose for NFSv3 Gateway machine (referred to as the NFS server from here on):

```
$ sudo apt-get install nfs-common
```

Configuring the NFSv3 Gateway

Proceed as follows to configure the gateway.

1. Add the following property to `hdfs-site.xml` on the *NameNode*:

```
<property>
  <name>dfs.namenode.accesstime.precision</name>
  <value>3600000</value>
  <description>The access time for an HDFS file is precise up to this value. The
  default value is 1 hour.
  Setting a value of 0 disables access times for HDFS.</description>
</property>
```

2. Add the following property to `hdfs-site.xml` on the *NFS server*:

```
<property>
  <name>dfs.nfs3.dump.dir</name>
  <value>/tmp/.hdfs-nfs</value>
</property>
```



Note:

You should change the location of the file dump directory, which temporarily saves out-of-order writes before writing them to HDFS. This directory is needed because the NFS client often reorders writes, and so sequential writes can arrive at the NFS gateway in random order and need to be saved until they can be ordered correctly. After these out-of-order writes have exceeded 1MB in memory for any given file, they are dumped to the `dfs.nfs3.dump.dir` (the memory threshold is not currently configurable).

Make sure the directory you choose has enough space. For example, if an application uploads 10 files of 100MB each, `dfs.nfs3.dump.dir` should have roughly 1GB of free space to allow for a worst-case reordering of writes to every file.

3. Configure the user running the gateway (normally the `hdfs` user as in this example) to be a proxy for other users. To allow the `hdfs` user to be a proxy for all other users, add the following entries to `core-site.xml` on the *NameNode*:

```
<property>
  <name>hadoop.proxyuser.hdfs.groups</name>
  <value>*</value>
  <description>
    Set this to '*' to allow the gateway user to proxy any group.
  </description>
</property>
<property>
  <name>hadoop.proxyuser.hdfs.hosts</name>
  <value>*</value>
  <description>
    Set this to '*' to allow requests from any hosts to be proxied.
</description>
</property>
```



```
</description>
</property>
```

4. Restart the NameNode.

Starting the NFSv3 Gateway

Do the following on the NFS server.

1. First, stop the default NFS services, if they are running:

```
$ sudo service nfs stop
```

2. Start the HDFS-specific services:

```
$ sudo service hadoop-hdfs-nfs3 start
```

Verifying that the NFSv3 Gateway is Working

To verify that the NFS services are running properly, you can use the `rpcinfo` command on any host on the local network:

```
$ rpcinfo -p <nfs_server_ip_address>
```

You should see output such as the following:

```
program    vers    proto    port
100005     1       tcp      4242    mountd
100005     2       udp      4242    mountd
100005     2       tcp      4242    mountd
100000     2       tcp      111     portmapper
100000     2       udp      111     portmapper
100005     3       udp      4242    mountd
100005     1       udp      4242    mountd
100003     3       tcp      2049    nfs
100005     3       tcp      4242    mountd
```

To verify that the HDFS namespace is exported and can be mounted, use the `showmount` command.

```
$ showmount -e <nfs_server_ip_address>
```

You should see output similar to the following:

```
Exports list on <nfs_server_ip_address>:
/ (everyone)
```

Mounting HDFS on an NFS Client

To import the HDFS file system on an NFS client, use a `mount` command such as the following on the client:

```
$ mount -t nfs -o vers=3,proto=tcp,nolock <nfs_server_hostname>:/ /hdfs_nfs_mount
```



Note:

When you create a file or directory as user `hdfs` on the client (that is, in the HDFS file system imported using the NFS mount), the ownership may differ from what it would be if you had created it in HDFS directly. For example, ownership of a file created on the client might be `hdfs:hdfs` when the same operation done natively in HDFS resulted in `hdfs:supergroup`. This is because in native HDFS, BSD semantics determine the group ownership of a newly-created file: it is set to the same group as the parent directory where the file is created. When the operation is done over NFS, the typical Linux semantics create the file with the group of the effective GID (group ID) of the process creating the file, and this characteristic is explicitly passed to the NFS gateway and HDFS.

Setting HDFS Quotas

You can set quotas in HDFS for:

- The number of file and directory names used
- The amount of space used by given directories

Points to note:

- The quotas for names and the quotas for space are independent of each other.
- File and directory creation fails if the creation would cause the quota to be exceeded.
- The Reports Manager must index a file or directory before you can set a quota for it.
- Allocation fails if the quota would prevent a full block from being written; keep this in mind if you are using a large block size.
- If you are using replication, remember that each replica of a block counts against the quota.

About file count limits

- The file count quota is a limit on the number of file and directory names in the directory configured.
- A directory counts against its own quota, so a quota of 1 forces the directory to remain empty.
- File counts are based on the intended replication factor for the files; changing the replication factor for a file will credit or debit quotas.

About disk space limits

- The space quota is a hard limit on the number of bytes used by files in the tree rooted at the directory being configured.
- Each replica of a block counts against the quota.
- The disk space quota calculation takes replication into account, so it uses the replicated size of each file, not the user-facing size.
- The disk space quota calculation includes open files (files presently being written), as well as files already written.
- Block allocations for files being written will fail if the quota would not allow a full block to be written.

Setting HDFS Quotas Using Cloudera Manager

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

1. From the HDFS service page, select the **File Browser** tab.
2. Browse the file system to find the directory for which you want to set quotas.
3. Click the directory name so that it appears in the gray panel above the listing of its contents and in the detail section to the right of the File Browser table.
4. Click the **Edit Quota** button for the directory. A **Manage Quota** pop-up displays, where you can set file count or disk space limits for the directory you have selected.
5. When you have set the limits you want, click **OK**.

Setting HDFS Quotas Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

To set space quotas on a directory:

```
$ sudo -u hdfs hdfs dfsadmin -setSpaceQuota n directory
```

where *n* is a number of bytes and *directory* is the directory the quota applies to. You can specify multiple directories in a single command; *n* applies to each.

To remove space quotas from a directory:

```
$ sudo -u hdfs hdfs dfsadmin -clrSpaceQuota directory
```

You can specify multiple directories in a single command.

To set name quotas on a directory:

```
$ sudo -u hdfs hdfs dfsadmin -setQuota n directory
```

where *n* is the number of file and directory names in *directory*. You can specify multiple directories in a single command; *n* applies to each.

To remove name quotas from a directory:

```
$ sudo -u hdfs hdfs dfsadmin -clrQuota directory
```

You can specify multiple directories in a single command.

For More Information

For more information, see the [HDFS Quotas Guide](#).

Configuring Mountable HDFS

CDH 5 includes a FUSE (Filesystem in Userspace) interface into HDFS. The `hadoop-hdfs-fuse` package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. Proceed as follows.



Note: FUSE does not currently support file append operations.

Before you start: You must have a working HDFS cluster and know the hostname and port that your NameNode exposes.

To install hadoop-hdfs-fuses On Red Hat-compatible systems:

```
$ sudo yum install hadoop-hdfs-fuse
```

To install hadoop-hdfs-fuse on Ubuntu systems:

```
$ sudo apt-get install hadoop-hdfs-fuse
```

To install hadoop-hdfs-fuse on SLES systems:

```
$ sudo zypper install hadoop-hdfs-fuse
```

You now have everything you need to begin mounting HDFS on Linux.

To set up and test your mount point in a non-HA installation:

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_point>
```

where *namenode_port* is the NameNode's RPC port, `dfs.namenode.servicerpc-address`.

To set up and test your mount point in an HA installation:

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs dfs://<nameservice_id> <mount_point>
```

where *nameservice_id* is the value of `fs.defaultFS`. In this case the port defined for `dfs.namenode.rpc-address.[nameservice ID].[name node ID]` is used automatically. See [Enabling HDFS HA](#) on page 288 for more information about these properties.

You can now run operations as if they are on your mount point. Press **Ctrl+C** to end the `fuse-dfs` program, and unmount the partition if it is still mounted.



Note:

To find its configuration directory, `hadoop-fuse-dfs` uses the `HADOOP_CONF_DIR` configured at the time the `mount` command is invoked.

To clean up your test:

```
$ umount <mount_point>
```

You can now add a permanent HDFS mount which persists through reboots.

To add a system mount:

1. Open `/etc/fstab` and add lines to the bottom similar to these:

```
hadoop-fuse-dfs#dfs://<name_node_hostname>:<namenode_port> <mount_point> fuse
allow_other,usetrash,rw 2 0
```

For example:

```
hadoop-fuse-dfs#dfs://localhost:8020 /mnt/hdfs fuse allow_other,usetrash,rw 2 0
```



Note:

In an HA deployment, use the HDFS nameservice instead of the NameNode URI; that is, use the value of `dfs.nameservices` in `hdfs-site.xml`.

2. Test to make sure everything is working properly:

```
$ mount <mount_point>
```

Your system is now configured to allow you to use the `ls` command and use that mount point as if it were a normal system disk.

For more information, see the help for `hadoop-fuse-dfs`:

```
$ hadoop-fuse-dfs --help
```

Optimizing Mountable HDFS

- We recommend that you use the `-obig_writes` option on kernels later than 2.6.26. This option allows for better performance of writes.
- By default, the CDH 5 package installation creates the `/etc/default/hadoop-fuse` file with a maximum heap size of 128 MB. You might need to change the JVM minimum and maximum heap size for better performance. For example:

```
export LIBHDFS_OPTS="-Xms64m -Xmx256m"
```

Be careful not to set the minimum to a higher value than the maximum.

Configuring Centralized Cache Management in HDFS

Centralized cache management in HDFS is an explicit caching mechanism that allows users to specify paths to be cached by HDFS. The NameNode will communicate with DataNodes that have the desired blocks on disk, and instruct them to cache the blocks in off-heap caches.

This has several advantages:

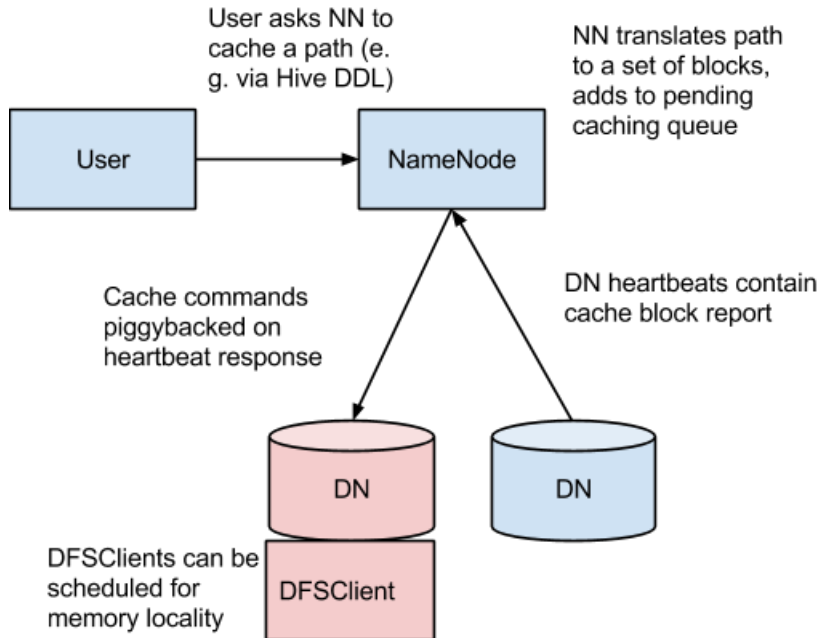
- Explicit pinning prevents frequently used data from being evicted from memory. This is particularly important when the size of the working set exceeds the size of main memory, which is common for many HDFS workloads.
- Since DataNode caches are managed by the NameNode, applications can query the set of cached block locations when making task placement decisions. Co-locating a task with a cached block replica improves read performance.
- When block has been cached by a DataNode, clients can use a new, more-efficient, zero-copy read API. Since checksum verification of cached data is done once by the DataNode, clients can incur essentially zero overhead when using this new API.
- Centralized caching can improve overall cluster memory utilization. When relying on the OS buffer cache at each DataNode, repeated reads of a block will result in all n replicas of the block being pulled into buffer cache. With centralized cache management, you can explicitly pin only m of the n replicas, saving $n-m$ memory.

Use Cases

Centralized cache management is most useful for files that are accessed repeatedly. For example, a fact table in Hive which is often used in joins is a good candidate for caching. On the other hand, caching the input of a one-year reporting query is probably less useful, since the historical data might be read only once.

Centralized cache management is also useful for mixed workloads with performance SLAs. Caching the working set of a high-priority workload insures that it does not contend for disk I/O with a low-priority workload.

Architecture



In this architecture, the NameNode is responsible for coordinating all the DataNode off-heap caches in the cluster. The NameNode periodically receives a "cache report" from each DataNode which describes all the blocks cached on a given DataNode. The NameNode manages DataNode caches by piggybacking cache and uncache commands on the DataNode heartbeat.

The NameNode queries its set of cache directives to determine which paths should be cached. Cache directives are persistently stored in the fsimage and edit log, and can be added, removed, and modified using Java and command-line APIs. The NameNode also stores a set of cache pools, which are administrative entities used to group cache directives together for resource management and enforcing permissions.

The NameNode periodically rescans the namespace and active cache directories to determine which blocks need to be cached or uncached and assign caching to DataNodes. Rescans can also be triggered by user actions like adding or removing a cache directive or removing a cache pool.

We do not currently cache blocks which are under construction, corrupt, or otherwise incomplete. If a cache directive covers a symlink, the symlink target is not cached. Caching is currently done on a per-file basis, although we would like to add block-level granularity in the future.

Concepts

Cache Directive

A **cache directive** defines a path that should be cached. Paths can be either directories or files. Directories are cached non-recursively, meaning only files in the first-level listing of the directory will be cached. Directives also specify additional parameters, such as the cache replication factor and expiration time. The replication factor specifies the number of block replicas to cache. If multiple cache directives refer to the same file, the maximum cache replication factor is applied.

The expiration time is specified on the command line as a `time-to-live` (TTL), a relative expiration time in the future. After a cache directive expires, it is no longer considered by the NameNode when making caching decisions.

Cache Pool

A **cache pool** is an administrative entity used to manage groups of cache directives. Cache pools have UNIX-like permissions that restrict which users and groups have access to the pool. Write permissions allow users to add and remove cache directives to the pool. Read permissions allow users to list the cache directives in a pool, as well as additional metadata. Execute permissions are unused.

Cache pools are also used for resource management. Pools can enforce a maximum `limit`, which restricts the number of bytes that can be cached in aggregate by directives in the pool. Normally, the sum of the pool limits will approximately equal the amount of aggregate memory reserved for HDFS caching on the cluster. Cache pools also track a number of statistics to help cluster users determine what is and should be cached.

Pools also enforce a maximum time-to-live. This restricts the maximum expiration time of directives being added to the pool.

cacheadmin Command-Line Interface

On the command-line, administrators and users can interact with cache pools and directives using the `hdfs cacheadmin` subcommand. Cache directives are identified by a unique, non-repeating 64-bit integer ID. IDs are not reused even if a cache directive is later removed. Cache pools are identified by a unique string name.

Cache Directive Commands

addDirective

Description: Add a new cache directive.

Usage: `hdfs cacheadmin -addDirective -path <path> -pool <pool-name> [-force] [-replication <replication>] [-ttl <time-to-live>]`

Where, `path`: A path to cache. The path can be a directory or a file.

`pool-name`: The pool to which the directive will be added. You must have write permission on the cache pool in order to add new directives.

`force`: Skips checking of cache pool resource limits.

`replication`: The cache replication factor to use. Defaults to 1.

`time-to-live`: Time period for which the directive is valid. Can be specified in seconds, minutes, hours, and days, for example: 30m, 4h, 2d. The value `never` indicates a directive that never expires. If unspecified, the directive never expires.

removeDirective

Description: Remove a cache directive.

Usage: `hdfs cacheadmin -removeDirective <id>`

Where, `id`: The id of the cache directive to remove. You must have write permission on the pool of the directive in order to remove it. To see a list of PathBasedCache directive IDs, use the `-listDirectives` command.

removeDirectives

Description: Remove every cache directive with the specified path.

Usage: `hdfs cacheadmin -removeDirectives <path>`

Where, `path`: The path of the cache directives to remove. You must have write permission on the pool of the directive in order to remove it.

listDirectives

Description: List PathBasedCache directives.

Usage: `hdfs cacheadmin -listDirectives [-stats] [-path <path>] [-pool <pool>]`

Where, `path`: List only PathBasedCache directives with this path. Note that if there is a PathBasedCache directive for `path` in a cache pool that we do not have read access for, it will not be listed.

`pool`: List only path cache directives in that pool.

`stats`: List path-based cache directive statistics.

Cache Pool Commands

addPool

Description: Add a new cache pool.

Usage: `hdfs cacheadmin -addPool <name> [-owner <owner>] [-group <group>] [-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]`

Where, `name`: Name of the new pool.

`owner`: Username of the owner of the pool. Defaults to the current user.

`group`: Group of the pool. Defaults to the primary group name of the current user.

`mode`: UNIX-style permissions for the pool. Permissions are specified in octal, for example: 0755. By default, this is set to 0755.

`limit`: The maximum number of bytes that can be cached by directives in this pool, in aggregate. By default, no limit is set.

`maxTtl`: The maximum allowed time-to-live for directives being added to the pool. This can be specified in seconds, minutes, hours, and days, for example: 120s, 30m, 4h, 2d. By default, no maximum is set. A value of `never` specifies that there is no limit.

modifyPool

Description: Modify the metadata of an existing cache pool.

Usage: `hdfs cacheadmin -modifyPool <name> [-owner <owner>] [-group <group>] [-mode <mode>] [-limit <limit>] [-maxTtl <maxTtl>]`

Where, `name`: Name of the pool to modify.

`owner`: Username of the owner of the pool.

`group`: Groupname of the group of the pool.

`mode`: Unix-style permissions of the pool in octal.

`limit`: Maximum number of bytes that can be cached by this pool.

`maxTtl`: The maximum allowed time-to-live for directives being added to the pool.

removePool

Description: Remove a cache pool. This also uncaches paths associated with the pool.

Usage: `hdfs cacheadmin -removePool <name>`

Where, `name`: Name of the cache pool to remove.

listPools

Description: Display information about one or more cache pools, for example: name, owner, group, permissions, and so on.

Usage: `hdfs cacheadmin -listPools [-stats] [<name>]`

Where, `name`: If specified, list only the named cache pool.

`stats`: Display additional cache pool statistics.

help

Description: Get detailed help about a command.

Usage: `hdfs cacheadmin -help <command-name>`

Where, `command-name`: The command for which to get detailed help. If no command is specified, print detailed help for all commands.

Configuration

Native Libraries

To lock block files into memory, the DataNode relies on native JNI code found in `libhadoop.so`. Be sure to [enable JNI](#) if you are using HDFS centralized cache management.

Configuration Properties

Required

Be sure to configure the following in `/etc/default/hadoop/conf/hdfs-default.xml`:

- `dfs.datanode.max.locked.memory`: The maximum amount of memory a DataNode will use for caching (in bytes). The "locked-in-memory size" `ulimit (ulimit -l)` of the DataNode user also needs to be increased to match this parameter (see [OS Limits](#)). When setting this value, remember that you will need space in memory for other things as well, such as the DataNode and application JVM heaps and the operating system page cache.

Optional

The following properties are not required, but may be specified for tuning:

- `dfs.namenode.path.based.cache.refresh.interval.ms`: The NameNode uses this as the amount of milliseconds between subsequent path cache rescans. This calculates the blocks to cache and each DataNode containing a replica of the block that should cache it. By default, this parameter is set to 300000, which is five minutes.
- `dfs.datanode.fsdatasetcache.max.threads.per.volume`: The DataNode uses this as the maximum number of threads per volume to use for caching new data. By default, this parameter is set to 4.
- `dfs.cachereport.intervalMsec`: The DataNode uses this as the amount of milliseconds between sending a full report of its cache state to the NameNode. By default, this parameter is set to 10000, which is 10 seconds.
- `dfs.namenode.path.based.cache.block.map.allocation.percent`: The percentage of the Java heap which we will allocate to the cached blocks map. The cached blocks map is a hash map which uses chained hashing. Smaller maps may be accessed more slowly if the number of cached blocks is large; larger maps will consume more memory. By default, this parameter is set to 0.25 percent.

OS Limits

If you get the error `Cannot start datanode because the configured max locked memory size... is more than the datanode's available RLIMIT_MEMLOCK ulimit`, that means that the operating system is imposing a lower limit on the amount of memory that you can lock than what you have configured. To fix this, you must adjust the `ulimit -l` value that the DataNode runs with. Usually, this value is configured in `/etc/security/limits.conf`. However, it will vary depending on what operating system and distribution you are using.

You will know that you have correctly configured this value when you can run `ulimit -l` from the shell and get back either a higher value than what you have configured with `dfs.datanode.max.locked.memory`, or the string `unlimited`, indicating that there is no limit. Note that it's typical for `ulimit -l` to output the memory lock limit in KB, but `dfs.datanode.max.locked.memory` must be specified in bytes.

Managing Hive

Use the following procedures to manage HiveServer2 and the Hive metastore. To configure high availability for the Hive metastore, see [Hive Metastore High Availability](#) on page 338.

Heap Size and Garbage Collection for Hive Components

Hive Component Memory Recommendations

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

Number of Concurrent Connections	HiveServer2 Heap Size Minimum Recommendation	Hive Metastore Heap Size Minimum Recommendation
Up to 40 concurrent connections (Cloudera recommends splitting HiveServer2 into multiple instances and load balancing once you start allocating >12 GB to HiveServer2. The objective is to size to reduce impact of Java garbage collection on active processing by the service.	12 GB	12 GB
Up to 20 concurrent connections	6 GB	10 GB
Up to 10 concurrent connections	4 GB	8 GB
Single connection	2 GB	4 GB



Important: These numbers are general guidance only, and may be affected by factors such as number of columns, partitions, complex joins, and client activity among other things. It is important to review and refine through testing based on your anticipated deployment to arrive at best values for your environment.

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

The `permGenSize` should be set to 512M for all.

Configuring Heap Size and Garbage Collection for Hive Components

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 the `hive-env.sh` advanced configuration snippet if you use Cloudera Manager or otherwise edit `/etc/hive/hive-env.sh`.

To configure the heap size for the Beehive CLI, set the `HADOOP_HEAPSIZE` environment variable in the `hive-env.sh` advanced configuration snippet if you use Cloudera Manager or otherwise edit `/etc/hive/hive-env.sh` before starting the Beehive 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. Uncomment the lines by removing the # character.

```
if [ "$SERVICE" = "cli" ]; then
  if [ -z "$DEBUG" ]; then
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms10m
-XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:+UseParNewGC -XX:-UseGCOverheadLimit"
  else
    export HADOOP_OPTS="$HADOOP_OPTS -XX:NewRatio=12 -Xmx12288m -Xms10m
```

```
-XX:MaxHeapFreeRatio=40 -XX:MinHeapFreeRatio=15 -XX:-UseGCOverheadLimit"
fi
fi
export HADOOP_HEAPSIZE=2048
```

You can choose whether to use the Concurrent Collector or the New Parallel Collector for garbage collection, by passing `-XX:+UseParNewGC` or `-XX:+UseConcMarkSweepGC` in the `HADOOP_OPTS` lines above, and you can tune the garbage collection overhead limit by setting `-XX:-UseGCOverheadLimit`. To enable the garbage collection overhead limit, remove the setting or change it to `-XX:+UseGCOverheadLimit`.

Configuration for WebHCat

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

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

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`.

Managing Hive Using Cloudera Manager

There are two Hive service roles:

- Hive Metastore Server - manages the metastore process when Hive is configured with a remote metastore. You are strongly encouraged to read [Configuring the Hive Metastore \(CDH 4\)](#) or [Configuring the Hive Metastore \(CDH 5\)](#).
- HiveServer2 - supports a Thrift API tailored for JDBC and ODBC clients, Kerberos authentication, and multi-client concurrency. There is also a CLI for HiveServer2 named Beeline. Cloudera recommends that you deploy HiveServer2 whenever possible. You can use the original HiveServer, and run it concurrently with HiveServer2. However, Cloudera Manager does not manage HiveServer, so you must configure and manage it outside Cloudera Manager. See [HiveServer2 documentation \(CDH 4\)](#) or [HiveServer2 documentation \(CDH 5\)](#) for more information.

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.

The Hive Metastore Server

Cloudera recommends using a remote Hive metastore, especially for CDH 4.2 and later. Since the remote metastore is recommended, Cloudera Manager treats the Hive Metastore Server as a required role for all Hive services. Here are a couple key reasons why the remote metastore setup is advantageous, especially in production settings:

- The Hive metastore database password and JDBC drivers don't need to be shared with every Hive client; only the Hive Metastore Server does. Sharing passwords with many hosts is a security concern.
- You can control activity on the Hive metastore database. To stop all activity on the database, just stop the Hive Metastore Server. This makes it easy to perform tasks such as backup and upgrade, which require all Hive activity to stop.

Information about the initial configuration of a remote Hive metastore database with Cloudera Manager can be found at [Cloudera Manager and Managed Service Data Stores](#).

Considerations When Upgrading Cloudera Manager

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

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

Cloudera Manager provides a Hive configuration option to bypass the Hive Metastore Server. When this configuration is enabled, Hive clients, Hue, and Impala connect directly to the Hive metastore database. Prior to Cloudera Manager 4.5, Hue and Impala connected directly to the Hive metastore database, so the bypass mode is enabled by default when upgrading to Cloudera Manager 4.5 or later. This ensures that the upgrade does not disrupt your existing setup. You should plan to disable the bypass mode, especially when using CDH 4.2 or later. Using the Hive Metastore Server is the recommended configuration, and the WebHCat Server role requires the Hive Metastore Server to *not* be bypassed. To disable bypass mode, see [Disabling Bypass Mode](#) on page 156.

Cloudera Manager 4.5 or later also supports HiveServer2 with CDH 4.2. In CDH 4, HiveServer2 is not added by default, but can be added as a new role under the Hive service (see [Role Instances](#) on page 45). In CDH 5, HiveServer2 is a mandatory role.

Disabling Bypass Mode

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

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

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Select **Scope** > **HIVE service_name (Service-Wide)**
4. Select **Category** > **Advanced**.
5. Deselect the **Bypass Hive Metastore Server** property.
6. Click **Save Changes** to commit the changes.
7. Re-deploy Hive client configurations.
8. Restart Hive and any Hue or Impala services configured to use that Hive service.

Using Hive Gateways

Because the Hive service does not have worker roles, another mechanism is needed to enable the automatic propagation of client configurations to the other hosts in your cluster. Gateway roles fulfill this function. Gateways in fact aren't really roles and do not have state, but they act as indicators for where client configurations should be placed. Hive gateways are created by default when the Hive service is added.

Hive Table Statistics

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

If your cluster has Impala then you can use the Impala implementation to compute statistics. The Impala implementation to compute table statistics is available in CDH 5.0.0 or higher and in Impala version 1.2.2 or higher. The Impala implementation of `COMPUTE STATS` requires no setup steps and is preferred over the Hive implementation. See

[Overview of Table Statistics](#). If you are running an older version of Impala, you can collect statistics on a Hive table by running the following command from a Beeline client connected to HiveServer2:

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

Managing User-Defined Functions (UDFs) with HiveServer2

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

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

Blacklist for Built-in UDFs

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

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

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

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

Creating Permanent Functions

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



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

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

10 Restart the Hive service.

11 **With Sentry enabled** - Grant privileges on the JAR files to the roles that require access. Log in to Beeline as user `hive` and use the Hive SQL [GRANT](#) statement to do so. For example:

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

You must also grant privilege to the JAR on HDFS:

```
GRANT ALL ON URI 'hdfs:///path/to/jar' TO ROLE EXAMPLE_ROLE
```

12 Run the `CREATE FUNCTION` command to create the UDF from the JAR file and point to the JAR file location in HDFS. For example:

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

Creating Temporary Functions

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



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

2. In the Cloudera Manager Admin Console, go to the Hive service.

3. Click the **Configuration** tab.

4. Expand the **Hive (Service-Wide)** scope.

5. Click the **Advanced** category.

6. Configure the **Hive Auxiliary JARs Directory** property with the HiveServer2 host path and the Hive Metastore host path from Step 1, for example `/opt/local/hive/lib/`. Setting this property overwrites `hive.aux.jars.path`, even if this variable has been previously set in the HiveServer2 advanced configuration snippet.

7. Click **Save Changes**. The JARs are added to `HIVE_AUX_JARS_PATH` environment variable.

8. Redeploy the Hive client configuration.

a. In the Cloudera Manager Admin Console, go to the Hive service.

b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.

c. Click **Deploy Client Configuration**.

9. Restart the Hive service.

10 **With Sentry enabled** - Grant privileges on the JAR files to the roles that require access. Log in to Beeline as user `hive` and use the Hive SQL [GRANT](#) statement to do so. For example:

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

You must also grant privilege to the JAR on HDFS:

```
GRANT ALL ON URI 'hdfs:///path/to/jar' TO ROLE EXAMPLE_ROLE
```

11 Run the `CREATE TEMPORARY FUNCTION` command. For example:

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

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

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

Creating Permanent Functions

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

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

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

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

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

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

You must also grant privilege to the JAR on HDFS:

```
GRANT ALL ON URI 'hdfs:///path/to/jar' TO ROLE EXAMPLE_ROLE
```

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

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

7. Run the `CREATE FUNCTION` command and point to the JAR from Hive:

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

Creating Temporary Functions

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

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

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

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

4. **If Sentry is enabled** - Grant privileges on the local JAR files to the roles that require access. Login to Beeline as user `hive` and use the Hive SQL [GRANT](#) statement to do so. For example:

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

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

```
udf_r = server=server1->uri=file:///opt/local/hive/lib
```

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

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

Running Hive on Spark

This section explains how to set up Hive on Spark. It assumes that your cluster is managed by Cloudera Manager.



Important: Hive on Spark is included in CDH 5.4 but is not currently supported nor recommended for production use. To try this feature in CDH 5.4, use it in a test environment.

Configuring Hive on Spark



Important: Hive on Spark is included in CDH 5.4 but is not currently supported nor recommended for production use. To try this feature in CDH 5.4, use it in a test environment.

This topic explains the configuration properties you set up to run Hive on Spark.



Note: We recommend that you use HiveServer2 with Beeline. The following content, except for [Configuring Hive on Spark for Hive CLI](#) on page 163, is based on this assumption.

Installation Considerations

For Hive to work on Spark, you must deploy Spark gateway roles on the same machine that hosts HiveServer2. Otherwise, Hive on Spark cannot read from Spark configurations and cannot submit Spark jobs. For more information about gateway roles, see [Managing Roles](#) on page 44.

After installation, run the following command in Hive so that Hive will use Spark as the back-end engine for all subsequent queries.

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

Enabling Hive on Spark

By default, Hive on Spark is not enabled. To enable Hive on Spark, perform the following steps in Cloudera Manager.

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Enter `Enable Hive on Spark` in the **Search** field.
4. Check the box for **Enable Hive on Spark (Unsupported)**.
5. Locate the **Spark On YARN Service** and click **SPARK_ON_YARN**.
6. Click **Save Changes** to commit the changes.

Configuration Properties

Property	Description
<code>hive.stats.collect.rawdatasize</code>	<p>Hive on Spark uses statistics to determine the threshold for converting common join to map join. There are two types of statistics about the size of data:</p> <ul style="list-style-type: none"> <code>totalSize</code>: The approximate size of data on the disk <code>rawDataSize</code>: The approximate size of data in memory <p>When both metrics are available, Hive chooses <code>rawDataSize</code>.</p> <p>Default: <code>True</code></p>
<code>hive.auto.convert.join.noconditionaltask.size</code>	<p>The threshold for the sum of all the small table size (by default, <code>rawDataSize</code>), for map join conversion. You can increase the value if you want better performance by converting more common joins to map joins. However, if you set this value too high, tasks may fail because too much memory is being used by data from small tables.</p> <p>Default: <code>20MB</code></p>

Configuring Hive

For improved performance, Cloudera recommends that you configure the following additional properties for Hive. Set these properties in Cloudera Manager Safety Valve for HiveServer2.

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

Configuring Spark

Configure the following Spark properties to suit your cluster environment. During initial deployment, rules in Cloudera Manager tune this according to your cluster environment.

Property	Description
<code>spark.executor.cores</code>	The number of cores per Spark executor.
<code>spark.executor.memory</code>	The maximum size of each Spark executor's Java heap memory when Hive is running on Spark.
<code>spark.yarn.executor.memoryOverhead</code>	The amount of extra off-heap memory that can be requested from YARN, per executor process. Combined with <code>spark.executor.memory</code> , this is the total memory YARN can use to create a JVM for an executor process.
<code>spark.driver.memory</code>	The maximum size of each Spark driver's Java heap memory when Hive is running on Spark.
<code>spark.yarn.driver.memoryOverhead</code>	The amount of extra off-heap memory that can be requested from YARN per driver. Combined with <code>spark.driver.memory</code> , this is the total memory that YARN can use to create a JVM for a driver process.

Enabling Spark Executor Allocation

Spark can dynamically scale the set of cluster resources allocated to your application up and down, based on the workload. Dynamic allocation is useful when multiple applications share resources in a Spark cluster. When an application becomes idle, its resources can be released to the resource pool and acquired by other applications. Cloudera recommends that you enable dynamic allocation by setting `spark.executor.dynamicAllocation.enabled` to `true`. This is the default value in Cloudera Manager.

When you enable dynamic allocation, Spark adds and removes executors dynamically to Hive jobs, based on workload. The following table describes additional properties.

Property	Description
<code>spark.executor.dynamicAllocation.initialExecutors</code>	The initial number of executors for a Spark application when dynamic allocation is enabled. The default is 1.
<code>spark.executor.dynamicAllocation.minExecutors</code>	The lower bound for the number of executors. The default is 1.
<code>spark.executor.dynamicAllocation.maxExecutors</code>	The upper bound for the number of executors. The default is <code>Integer.MAX_VALUE</code> .

If you disable dynamic scaling, configure the following property:

Property	Description
<code>spark.executor.instances</code>	The total number of executors used for the Spark application.

Configuring Executor Memory Size

Executor memory size can have a number of effects on Hive. Increasing executor memory increases the number of queries for which Hive can enable mapjoin optimization. However, if there's too much executor memory, it takes longer to perform garbage collection. Also, some experiments shows that HDFS doesn't handle concurrent writers well, so it may face a race condition if there are too many executor cores.

Cloudera recommends that you set the value for `spark.executor.cores` to 5, 6 or 7, depending on what the host is divisible by. For example, if `yarn.nodemanager.resource.cpu-vcores` is 19, then you would set the value to 6. Executors must have the same number of cores. If you set the value to 5, each executor only gets three cores, with four left unused. If you set the value to 7, only two executors are used, and five cores are unused. If the number of cores is 20, set the value to 5 so that each executor gets four cores, and no cores are unused.

Cloudera also recommends the following:

- Compute a memory size equal to `yarn.nodemanager.resource.memory-mb * (spark.executor.cores / yarn.nodemanager.resource.cpu-vcores)` and then split that between `spark.executor.memory` and `spark.yarn.executor.memoryOverhead`.
- `spark.yarn.executor.memoryOverhead` is 15-20% of the total memory size.

Troubleshooting Hive on Spark



Important: Hive on Spark is included in CDH 5.4 but is not currently supported nor recommended for production use. To try this feature in CDH 5.4, use it in a test environment.

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

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. The query waits for YARN containers to initialize. Subsequent queries will be faster.

Problem: Exception Error: org.apache.thrift.transport.TTransportException (state=08S01,code=0) and HiveServer2 is down

HiveServer2 memory is set too small. For more information, see `STDOUT` for HiveServer2. To fix this issue:

1. In Cloudera Manager, go to **HIVE**.
2. Click **Configuration**.
3. Search for `Java Heap Size of HiveServer2 in Bytes`, and change it to be a larger value. Cloudera recommends a minimum value of 256 MB.
4. Restart HiveServer2.

Problem: Out-of-memory error

You might get 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
```

This error indicates that the Spark driver does not have enough off-heap memory. Increase the off-heap memory by setting `spark.yarn.driver.memoryOverhead` or `spark.driver.memory`.

Problem: Hive on Spark does not work with HBase

Hive on Spark with HBase is not supported. If you use HBase, use Hive on MapReduce instead of Hive on Spark.

Problem: Spark applications stay alive forever and occupy cluster resources

This can occur if there are multiple concurrent Hive sessions. To manually terminate the Spark applications:

1. Find the YARN application IDs for the applications by going to Cloudera Manager and clicking **Yarn > ResourceManager > ResourceManager Web UI**.
2. Log in to the YARN ResourceManager host.
3. Open a terminal and run:

```
yarn application -kill <applicationID>
```

`applicationID` is each YARN application ID you found in step 1.

Configuring Hive on Spark for Hive CLI



Important: Hive on Spark is included in CDH 5.4 but is not currently supported nor recommended for production use. To try this feature in CDH 5.4, use it in a test environment.

To use the Hive CLI, perform the following tasks in Cloudera Manager.

Tip: Cloudera recommends using HiveServer2 with Beeline. All Cloudera Manager–related configuration properties for Hive on Spark are for the HiveServer2 role. Also, the Hive CLI cannot read from `spark-defaults.conf`. As a result, the Hive CLI could perform poorly.

1. Go to the Hive service.
2. Click **HiveServer2** in the **Status Summary** section.
3. On the **HiveServer2** page, click the **Processes** tab.
4. In the **Configuration Files** section, click **hive-site.xml**.

Cloudera Manager opens the file in a new tab.

5. Go to the new tab and copy all properties in the `hive-site.xml` file.
6. Go back to the tab with the Cloudera Manager screen and click the breadcrumb link to go back to the Hive service page.
7. Click the **Configuration** tab.
8. Enter `hive-site` in the **Search** field.

9. In the **Hive Client Advanced Configuration Snippet (Safety Valve)** text field, paste the content from `hive-site.xml` file. Only keep the following properties:

- `hive.auto.convert.join`
- `hive.auto.convert.join.noconditionaltask.size`
- `hive.optimize.bucketmapjoin.sortedmerge`
- `hive.smbjoin.cache.rows`
- `hive.exec.reducers.max`
- `hive.vectorized.groupby.checkinterval`
- `hive.vectorized.groupby.flush.percent`
- `hive.compute.query.using.stats`
- `hive.vectorized.execution.enabled`
- `hive.vectorized.execution.reduce.enabled`
- `hive.merge.mapfiles`
- `hive.merge.mapredfiles`
- `hive.cbo.enable`
- `hive.fetch.task.conversion`
- `hive.fetch.task.conversion.threshold`
- `hive.limit.pushdown.memory.usage`
- `hive.merge.sparkfiles`
- `hive.merge.smallfiles.avgsize`
- `hive.merge.size.per.task`
- `hive.optimize.reducededuplication`
- `hive.optimize.reducededuplication.min.reducer`
- `hive.map.aggr`
- `hive.map.aggr.hash.percentmemory`
- `hive.optimize.sort.dynamic.partition`
- `spark.executor.memory`
- `spark.driver.memory`
- `spark.executor.cores`
- `spark.master`
- `spark.yarn.driver.memoryOverhead`
- `spark.yarn.executor.memoryOverhead`
- `spark.dynamicAllocation.enabled`
- `spark.dynamicAllocation.minExecutors`
- `spark.dynamicAllocation.initialExecutors`
- `hive.entity.capture.input.URI`
- `spark.shuffle.service.enabled`

10 In the **Search** field, enter `hive-env`.

11 In the **Gateway Client Environment Advanced Configuration Snippet for hive-env.sh (Safety Valve)** field, enter `AUX_CLASSPATH=${AUX_CLASSPATH}:/etc/spark/conf`.

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

13 Click **Actions > Deploy Client Configuration**.



Important: When using Hive on Spark with Hive CLI, Hive Clients require Spark on YARN client configuration. Every host with a Hive role must also have a Spark on YARN Gateway. Deploy Spark client configurations whenever there's a change in order for Hive clients to pick up the change.

Managing Hue

Hue is a set of web UIs that enable you to interact with a CDH cluster. This section describes tasks for managing Hue.

Adding a Hue Service and Role Instance
 Adding the Hue Service

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

After initial installation, you can use the **Add a Service** wizard to add and configure a new Hue service instance.

1. On the Home page, click



to the right of the cluster name and select **Add a Service**. A list of service types display.

2. Select **Hue**.
3. Click **Continue**.

A page displays where you can specify the dependencies for the Hue service.

4. Select the row with the Hue dependencies required for your cluster. For more information, see [Hue Dependencies](#).
5. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

6. Click **Continue**.

Cloudera Manager starts the Hue service.

7. Click **Continue**.
8. Click **Finish**.
9. If your cluster uses Kerberos, Cloudera Manager will automatically add a **Hue Kerberos Ticket Renewer** role to each host where you assigned the Hue Server role instance. Also see, [Enable Hue to Work with Hadoop Security using Cloudera Manager](#).

Adding a Hue Role Instance

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

1. In Cloudera Manager Administration Console, go to the Hue service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. If your cluster uses Kerberos, you must add the **Hue Kerberos Ticket Renewer** role to each host where you assigned the Hue Server role instance. Cloudera Manager will throw a validation error if the new Hue Server role does not have a colocated KT Renewer role. Also see, [Enable Hue to Work with Hadoop Security using Cloudera Manager](#).
6. Click **Continue**.

Hue and High Availability

If your cluster has HDFS high availability enabled, you must configure the Hue HDFS Web Interface Role property to use HttpFS. See [Configuring Hue to Work with HDFS HA](#) on page 304 for detailed instructions.

To configure the Hue service itself for high availability, see [Hue High Availability](#) on page 340.

Managing Hue Analytics Data Collection

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

Hue tracks anonymized pages and application versions to collect information used to compare each application's usage levels. The data collected does not include hostnames or IDs; For example, the data has the format /2.3.0/pig, /2.5.0/beeswax/execute. You can restrict data collection as follows:

1. Go to the Hue service.
2. Click the **Configuration** tab.
3. Select **Scope > Hue**.
4. Locate the **Enable Usage Data Collection** property or search for it by typing its name in the Search box.
5. Deselect the **Enable Usage Data Collection** checkbox.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Restart the Hue service.

Enabling Hue Applications Using Cloudera Manager

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

Most Hue applications are configured by default, based on the services you have installed. Cloudera Manager selects the service instance that Hue depends on. If you have more than one service, you may want to verify or change the service dependency for Hue. Also, if you add a service such as Sqoop 2 or Oozie after you have set up Hue, you need to set the dependency because it is not done automatically. To add a dependency:

1. Go to the **Hue** service.
2. Click the **Configuration** tab.

3. Select **Scope** > **Hue (Service-Wide)**.
4. Select **Category** > **Main**.
5. Select each **service name Service** property to set the dependency. Select **none** to remove the dependency.
6. Click **Save Changes** to commit the changes.
7. Restart the Hue service.

Enabling the Sqoop 2 Application

If you are upgrading Cloudera Manager from a release 4.6 or lower, you need to set the Hue dependency to enable the Sqoop 2 application.

Enabling the HBase Browser Application with doAs Impersonation

Minimum Required Role: [Full Administrator](#)

The Hue HBase application communicates through a proxy server called the HBase Thrift Server, which then forwards commands to HBase. Because Hue stands between the Thrift server and the actual user, all HBase operations appear to come from the `hue` user and not the actual user. To secure these interactions, you must do the following:

- Ensure that users logged into Hue perform operations with their own privileges, and not those of the impersonating `hue` user.
- Once Hue can impersonate other users, ensure that *only* the Hue server can send commands to the HBase Thrift server. To ensure this, use Kerberos to authenticate the `hue` user to the HBase Thrift server.

To enable the HBase browser application:

1. [Add the HBase Thrift Server role](#).
2. If you have a [Kerberos-enabled](#) cluster, enable impersonation by configuring the following HBase properties:
 - a. Select the **HBase** service.
 - b. Click the **Configuration** tab.
 - c. Select **Scope** > **Service-Wide**.
 - d. Select **Category** > **Security**.
 - e. For the **HBase Thrift Authentication** property, make sure it is set to one of the following values:
 - `auth-conf`: authentication, integrity and confidentiality checking
 - `auth-int`: authentication and integrity checking
 - `auth`: authentication only
 - f. Select **Category** > **Main**.
 - g. Check the **Enable HBase Thrift Http Server** and **Enable HBase Thrift Proxy Users** properties checkboxes.
 - h. Click **Save Changes** to commit the changes.
3. Configure Hue to point to the Thrift Server and to a valid HBase configuration directory:
 - a. Select the **Hue** service.
 - b. Click the **Configuration** tab.
 - c. Select **Scope** > **All**.
 - d. Select **Category** > **Main**.
 - e. For the **HBase Service** property, make sure it is set to the HBase service for which you enabled the Thrift Server role (if you have more than one HBase service instance).
 - f. In the **HBase Thrift Server** property, click the edit field and select the Thrift Server role for Hue to use.
 - g. Select **Category** > **Advanced**.
 - h. Locate the **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini** property and add the following property:

```
[hbase]
hbase_conf_dir=/etc/hbase/conf
```

- i. Click **Save Changes** to commit the changes.

Enabling the Solr Search Application

To use the Solr Search application with Hue, you must update the URL for the Solr Server in the Hue Server advanced configuration snippet. In addition, if you are using parcels with CDH 4.3, you must register the "hue-search" application manually, or access will fail. See [Deploying Solr with Hue](#) on page 216 for detailed instructions.

Using an External Database for Hue

Cloudera strongly recommends an external database for clusters with multiple Hue users, especially clusters in a production environment. The default database, SQLite, works best with a single user and a small dataset. For supported databases, see:

- [CDH 4 supported databases](#)
- [CDH 5 supported databases](#)

Using an External Database for Hue Using Cloudera Manager

Minimum Required Role: [Full Administrator](#)

The Hue server requires an SQL database to store small amounts of data, including user account information and the history of job submissions and Hive queries. The Hue server supports a lightweight embedded database and several types of external databases. See the procedures below to configure Hue with an external database.



Warning: Cloudera strongly recommends an external database for clusters with multiple Hue users. See [Supported Databases](#).

In the instructions that follow, dumping the database and editing the JSON objects is only necessary if you have data in SQLite that you need to migrate. Otherwise, you can skip those steps.

Configuring the Hue Server to Store Data in MySQL



Note: Hue on CDH 5 requires InnoDB, *not* MyISAM, as the MySQL engine.

1. In the Cloudera Manager Admin Console, go to the Hue service status page.
2. Select **Actions > Stop**. Confirm you want to stop the service by clicking **Stop**.
3. Select **Actions > Dump Database**. Confirm you want to dump the database by clicking **Dump Database**.
4. Note the host to which the dump was written under **Step** in the **Dump Database Command** window. You can also find it by selecting **Commands > Recent Commands > Dump Database**.
5. Open a terminal window for the host and go to the dump file in `/tmp/hue_database_dump.json`.
6. Remove all JSON objects with `useradmin.userprofile` in the `model` field, for example:

```
{
  "pk": 14,
  "model": "useradmin.userprofile",
  "fields":
  {
    "creation_method": "EXTERNAL", "user": 14, "home_directory": "/user/tuser2" }
},
```

7. Set strict mode in `/etc/my.cnf` and restart MySQL:

```
[mysqld]
sql_mode=STRICT_ALL_TABLES
```


8. Create a new database and grant privileges to a Hue user to manage this database. For example:

```
mysql> create database hue CHARACTER SET utf8 COLLATE utf8_general_cs;
Query OK, 1 row affected (0.01 sec)
mysql> grant all on hue.* to 'hue'@'localhost' identified by 'secretpassword';
Query OK, 0 rows affected (0.00 sec)
```

9. In the Cloudera Manager Admin Console, click the Hue service.

10. Click the **Configuration** tab.

11. Select **Scope > All**.

12. Select **Category > Database**.

13. Specify the settings for **Hue Database Type**, **Hue Database Hostname**, **Hue Database Port**, **Hue Database Username**, **Hue Database Password**, and **Hue Database Name**. For example, for a MySQL database on the local host, you might use the following values:

- Hue Database Type = mysql
- Hue Database Hostname = *host*
- Hue Database Port = 3306
- Hue Database Username = hue
- Hue Database Password = *secretpassword*
- Hue Database Name = hue

14. Optionally restore the Hue data to the new database:

a. Select **Actions > Synchronize Database**.

b. Determine the foreign key ID.

```
$ mysql -uhue -psecretpassword
mysql > SHOW CREATE TABLE auth_permission;
```

c. **(InnoDB only)** Drop the foreign key that you retrieved in the previous step.

```
mysql > ALTER TABLE auth_permission DROP FOREIGN KEY content_type_id_refs_id_XXXXXX;
```

d. Delete the rows in the `django_content_type` table.

```
mysql > DELETE FROM hue.django_content_type;
```

e. In Hue service instance page, click **Actions > Load Database**. Confirm you want to load the database by clicking **Load Database**.

f. **(InnoDB only)** Add back the foreign key.

```
mysql > ALTER TABLE auth_permission ADD FOREIGN KEY (content_type_id) REFERENCES
django_content_type (id);
```

15. Start the Hue service.

Configuring the Hue Server to Store Data in PostgreSQL

1. In the Cloudera Manager Admin Console, go to the Hue service status page.

2. Select **Actions > Stop**. Confirm you want to stop the service by clicking **Stop**.

3. Select **Actions > Dump Database**. Confirm you want to dump the database by clicking **Dump Database**.

4. Note the host to which the dump was written under **Step** in the **Dump Database Command** window. You can also find it by selecting **Commands > Recent Commands > Dump Database**.

5. Open a terminal window for the host and go to the dump file in `/tmp/hue_database_dump.json`.

6. Remove all JSON objects with `useradmin.userprofile` in the `model` field, for example:

```
{
  "pk": 14,
  "model": "useradmin.userprofile",
  "fields":
  {
    "creation_method": "EXTERNAL", "user": 14, "home_directory": "/user/tuser2" }
},
```

7. Install required packages.

RHEL

```
$ sudo yum install postgresql-devel gcc python-devel
```

SLES

```
$ sudo zypper install postgresql-devel gcc python-devel
```

Ubuntu or Debian

```
$ sudo apt-get install postgresql-devel gcc python-devel
```

8. Install the Python module that provides the connector to PostgreSQL:

- **Parcel install**

```
$ sudo /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/pip install setuptools
$ sudo /opt/cloudera/parcels/CDH/lib/hue/build/env/bin/pip install psycopg2
```

- **Package install**

- **CDH 4**

```
sudo -u hue /usr/share/hue/build/env/bin/pip install setuptools
sudo -u hue /usr/share/hue/build/env/bin/pip install psycopg2
```

- **CDH 5**

```
sudo -u hue /usr/lib/hue/build/env/bin/pip install setuptools
sudo -u hue /usr/lib/hue/build/env/bin/pip install psycopg2
```

9. Install the PostgreSQL server.

RHEL

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

SLES

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

Ubuntu or Debian

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

10. Initialize the data directories.

```
$ service postgresql initdb
```

11. Configure client authentication.

- a. Edit `/var/lib/pgsql/data/pg_hba.conf`.
- b. Set the authentication methods for local to `trust` and for host to `password` and add the following line at the end.

```
host hue hue 0.0.0.0/0 md5
```

12 Start the PostgreSQL server.

```
$ su - postgres
# /usr/bin/postgres -D /var/lib/pgsql/data > logfile 2>&1 &
```

13 Configure PostgreSQL to listen on all network interfaces.

- a. Edit `/var/lib/pgsql/data/postgresql.conf` and set `listen_addresses`.

```
listen_addresses = '0.0.0.0'      # Listen on all addresses
```

14 Create the hue database and grant privileges to a hue user to manage the database.

```
# psql -U postgres
postgres=# create database hue;
postgres=# \c hue;
You are now connected to database 'hue'.
postgres=# create user hue with password 'secretpassword';
postgres=# grant all privileges on database hue to hue;
postgres=# \q
```

15 Restart the PostgreSQL server.

```
$ sudo service postgresql restart
```

16 Verify connectivity.

```
psql -h localhost -U hue -d hue
Password for user hue: secretpassword
```

17. Configure the PostgreSQL server to start at boot.

RHEL

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

SLES

```
$ sudo chkconfig --add postgresql
```

Ubuntu or Debian

```
$ sudo chkconfig postgresql on
```

- 18 In the Cloudera Manager Admin Console, click the Hue service.
- 19 Click the **Configuration** tab.
- 20 Select **Scope** > **All**.
- 21 Select **Category** > **Advanced**.
- 22 Specify the settings for **Hue Server Configuration Advanced Configuration Snippet**:



Note: If you specify the database host, port, username, password, and name in the respective **Service-Wide > Database > Hue Database *** properties, you can omit those properties from the configuration. In particular, you can avoid storing the password in the Hue configuration file in plain text.

```
[desktop]
[[database]]
host=localhost
port=5432
engine=postgresql_psycpg2
user=hue
password=secretpassword
name=hue
```

- 23 Click **Save Changes** to commit the changes.
- 24 Optionally restore the Hue data to the new database:
 - a. Select **Actions > Synchronize Database**.
 - b. Determine the foreign key ID.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# \d auth_permission;
```

- c. Drop the foreign key that you retrieved in the previous step.

```
postgres=# ALTER TABLE auth_permission DROP CONSTRAINT content_type_id_refs_id_XXXXXX;
```

- d. Delete the rows in the `django_content_type` table.

```
postgres=# TRUNCATE django_content_type CASCADE;
```

- e. In Hue service instance page, **Actions > Load Database**. Confirm you want to load the database by clicking **Load Database**.
- f. Add back the foreign key you dropped.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# ALTER TABLE auth_permission ADD CONSTRAINT content_type_id_refs_id_XXXXXX
FOREIGN KEY (content_type_id) REFERENCES django_content_type(id) DEFERRABLE INITIALLY
DEFERRED;
```

- 25 Start the Hue service.

Configuring the Hue Server to Store Data in Oracle (Parcel Installation)

Use the following instructions to configure the Hue Server with an Oracle database if you are working on a parcel-based deployment. If you are using packages, see [Configuring the Hue Server to Store Data in Oracle \(Package Installation\)](#) on page 174.



Important: Configure the database for character set `AL32UTF8` and national character set `UTF8`.

1. Install the required packages.

RHEL

```
$ sudo yum install gcc python-devel python-pip python-setuptools libaio
```

SLES:

Python devel packages are not included in SLES. Add the [SLES Software Development Kit \(SDK\)](#) as a repository and then install:

```
$ zypper install gcc libaio python-pip python-setuptools python-devel
```

Ubuntu or Debian

```
$ sudo apt-get install gcc python-dev python-pip python-setuptools libaio1
```

2. Download and add the [Oracle Client parcel](#) to the Cloudera Manager remote parcel repository URL list and download, distribute, and activate the parcel.
3. For CDH versions lower than 5.3, install the Python Oracle library:



Note: HUE_HOME is a reference to the location of your Hue installation. For package installs, this is usually `/usr/lib/hue`; for parcel installs, this is usually, `/opt/cloudera/parcels/<parcel version>/lib/hue/`.

```
$ HUE_HOME/build/env/bin/pip install cx_Oracle
```

4. For CDH versions lower than 5.3, upgrade django south:

```
$ HUE_HOME/build/env/bin/pip install south --upgrade
```

5. In the Cloudera Manager Admin Console, go to the Hue service status page.
6. Select **Actions > Stop**. Confirm you want to stop the service by clicking **Stop**.
7. Select **Actions > Dump Database**. Confirm you want to dump the database by clicking **Dump Database**.
8. Click the **Configuration** tab.
9. Select **Scope > All**.
10. Select **Category > Advanced**.
11. Set the **Hue Service Advanced Configuration Snippet (Safety Valve)** for `hue_safety_valve.ini` property.



Note: If you specify the database host, port, username, password, and name in the respective **Service-Wide > Database > Hue Database *** properties, you can omit those properties from the configuration. In particular, you can avoid storing the password in the Hue configuration file in plain text.

Add the following options (and modify accordingly for your setup):

```
[desktop]
[[database]]
host=localhost
port=1521
engine=oracle
user=hue
password=secretpassword
name=<SID of the Oracle database, for example, 'XE'>
```

For CDH 5.1 and higher you can use an Oracle service name. To use the Oracle service name instead of the SID, use the following configuration instead:

```
port=0
engine=oracle
user=hue
password=secretpassword
name=oracle.example.com:1521/orcl.example.com
```

The directive `port=0` allows Hue to use a service name. The `name` string is the connect string, including hostname, port, and service name.

To add support for a multithreaded environment, set the `threaded` option to `true` under the `[desktop]>[[database]]` section.

```
options={"threaded":true}
```

12 Grant required permissions to the `hue` user in Oracle:

```
grant alter any index to hue;
grant alter any table to hue;
grant alter database link to hue;
grant create any index to hue;
grant create any sequence to hue;
grant create database link to hue;
grant create session to hue;
grant create table to hue;
grant drop any sequence to hue;
grant select any dictionary to hue;
grant drop any table to hue;
grant create procedure to hue;
grant create trigger to hue;
```

13 Go to the Hue Server instance in Cloudera Manager and select **Actions > Synchronize Database**.

14 Ensure you are connected to Oracle as the `hue` user, then run the following command to delete all data from Oracle tables:

```
> set pagesize 100;
> SELECT 'DELETE FROM ' || table_name || ';' FROM user_tables;
```

15 Run the statements generated in the preceding step.

16 Commit your changes.

```
commit;
```

17 Load the data that you dumped. Go to the Hue Server instance and select **Actions > Load Database**. This step is not necessary if you have a fresh Hue install with no data or if you don't want to save the Hue data.

18 Start the Hue service.

Configuring the Hue Server to Store Data in Oracle (Package Installation)

If you have a parcel-based environment, see [Configuring the Hue Server to Store Data in Oracle \(Parcel Installation\)](#) on page 172.



Important: Configure the database for character set `AL32UTF8` and national character set `UTF8`.

1. Download the Oracle libraries at [Instant Client for Linux x86-64 Version 11.1.0.7.0](#), Basic and SDK (with headers) zip files to the same directory.
2. Unzip the Oracle client zip files.
3. Set environment variables to reference the libraries.

```
$ export ORACLE_HOME=oracle_download_directory
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME
```

4. Create a symbolic link for the shared object:

```
$ cd $ORACLE_HOME
$ ln -sf libclntsh.so.11.1 libclntsh.so
```

5. Install the required packages.

RHEL

```
$ sudo yum install gcc python-devel python-pip python-setuptools libaio
```

SLES:

Python devel packages are not included in SLES. Add the [SLES Software Development Kit \(SDK\)](#) as a repository and then install:

```
$ zypper install gcc libaio python-pip python-setuptools python-devel
```

Ubuntu or Debian

```
$ sudo apt-get install gcc python-dev python-pip python-setuptools libaio1
```

6. For CDH versions lower than 5.3, install the Python Oracle library:



Note: HUE_HOME is a reference to the location of your Hue installation. For package installs, this is usually /usr/lib/hue; for parcel installs, this is usually, /opt/cloudera/parcels/<parcel version>/lib/hue/.

```
$ HUE_HOME/build/env/bin/pip install cx_Oracle
```

7. For CDH versions lower than 5.3, upgrade django south:

```
$ HUE_HOME/build/env/bin/pip install south --upgrade
```

8. In the Cloudera Manager Admin Console, go to the Hue service status page.

9. Select **Actions > Stop**. Confirm you want to stop the service by clicking **Stop**.

10 Select **Actions > Dump Database**. Confirm you want to dump the database by clicking **Dump Database**.

11. Click the **Configuration** tab.

12 Select **Scope > All**.

13 Select **Category > Advanced**.

14 Set the **Hue Service Environment Advanced Configuration Snippet (Safety Valve)** property to

```
ORACLE_HOME=oracle_download_directory
LD_LIBRARY_PATH=$LD_LIBRARY_PATH:oracle_download_directory
```

15 Set the **Hue Service Advanced Configuration Snippet (Safety Valve)** for **hue_safety_valve.ini** property.



Note: If you specify the database host, port, username, password, and name in the respective **Service-Wide > Database > Hue Database *** properties, you can omit those properties from the configuration. In particular, you can avoid storing the password in the Hue configuration file in plain text.

Add the following options (and modify accordingly for your setup):

```
[desktop]
[[database]]
host=localhost
port=1521
engine=oracle
user=hue
password=secretpassword
name=<SID of the Oracle database, for example, 'XE'>
```

For CDH 5.1 and higher you can use an Oracle service name. To use the Oracle service name instead of the SID, use the following configuration instead:

```
port=0
engine=oracle
user=hue
password=secretpassword
name=oracle.example.com:1521/orcl.example.com
```

The directive `port=0` allows Hue to use a service name. The `name` string is the connect string, including hostname, port, and service name.

To add support for a multithreaded environment, set the `threaded` option to `true` under the `[desktop]>[[database]]` section.

```
options={"threaded":true}
```

16 Grant required permissions to the `hue` user in Oracle:

```
grant alter any index to hue;
grant alter any table to hue;
grant alter database link to hue;
grant create any index to hue;
grant create any sequence to hue;
grant create database link to hue;
grant create session to hue;
grant create table to hue;
grant drop any sequence to hue;
grant select any dictionary to hue;
grant drop any table to hue;
grant create procedure to hue;
grant create trigger to hue;
```

17 Go to the Hue Server instance in Cloudera Manager and select **Actions > Synchronize Database**.

18 Ensure you are connected to Oracle as the `hue` user, then run the following command to delete all data from Oracle tables:

```
> set pagesize 100;
> SELECT 'DELETE FROM ' || table_name || ';' FROM user_tables;
```

19 Run the statements generated in the preceding step.

20 Commit your changes.

```
commit;
```

21 Load the data that you dumped. Go to the Hue Server instance and select **Actions > Load Database**. This step is not necessary if you have a fresh Hue install with no data or if you don't want to save the Hue data.

22 Start the Hue service.

Using an External Database for Hue Using the Command Line

The Hue server requires an SQL database to store small amounts of data, including user account information and the history of job submissions and Hive queries. The Hue server supports a lightweight embedded database and several types of external databases. See the procedures below to configure Hue with an external database.



Warning: Cloudera strongly recommends an external database for clusters with multiple Hue users. See [Supported Databases](#).

Prerequisites

Before using an external database with Hue, install all of the support libraries required by your operating system. See [Development Preferences](#) in the Hue documentation for the full list.

Embedded Database

By default, Hue is configured to use the embedded database SQLite for this purpose, and should require no configuration or management by the administrator.

Inspecting the Embedded Hue Database

The default SQLite database used by Hue is located in `/var/lib/hue/desktop.db`. You can inspect this database from the command line using the `sqlite3` program. For example:

```
# sqlite3 /var/lib/hue/desktop.db
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select username from auth_user;
admin
test
sample
sqlite>
```



Important: It is strongly recommended that you avoid making any modifications to the database directly using `sqlite3`, though `sqlite3` is useful for management or troubleshooting.

Backing up the Embedded Hue Database

If you use the default embedded SQLite database, copy the `desktop.db` file to another node for backup. It is recommended that you back it up on a regular schedule, and also that you back it up before any upgrade to a new version of Hue.

External Database

Cloudera strongly recommends an external database for clusters with multiple Hue users, especially clusters in a production environment. The default database, SQLite, cannot support large data migrations. Hue supports MySQL, PostgreSQL, and Oracle. See [Supported Databases](#) for the supported versions.

In the instructions that follow, dumping the database and editing the JSON objects is only necessary if you have data in SQLite that you need to migrate. If you do not need to migrate data from SQLite, you can skip those steps.

Configuring the Hue Server to Store Data in MySQL



Note: Hue on CDH 5 requires InnoDB, *not* MyISAM, as the MySQL engine.

1. Shut down the Hue server if it is running.
2. Dump the existing database data to a text file. Note that using the `.json` extension is required.



Note: `HUE_HOME` is a reference to the location of your Hue installation. For package installs, this is usually `/usr/lib/hue`; for parcel installs, this is usually, `/opt/cloudera/parcels/<parcel version>/lib/hue/`.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field. Here are some examples of JSON objects that should be deleted.

```
{
  "pk": 1,
```

```

    "model": "useradmin.userprofile",
    "fields": {
      "creation_method": "HUE",
      "user": 1,
      "home_directory": "/user/alice"
    }
  },
  {
    "pk": 2,
    "model": "useradmin.userprofile",
    "fields": {
      "creation_method": "HUE",
      "user": 1100714,
      "home_directory": "/user/bob"
    }
  }
},
.....

```

4. Start the Hue server.
5. Install the MySQL client developer package.

OS	Command
RHEL	\$ sudo yum install mysql-devel
SLES	\$ sudo zypper install mysql-devel
Ubuntu or Debian	\$ sudo apt-get install libmysqlclient-dev

6. Install the MySQL connector.

OS	Command
RHEL	\$ sudo yum install mysql-connector-java
SLES	\$ sudo zypper install mysql-connector-java
Ubuntu or Debian	\$ sudo apt-get install libmysql-java

7. Install and start MySQL.

OS	Command
RHEL	\$ sudo yum install mysql-server
SLES	\$ sudo zypper install mysql \$ sudo zypper install libmysqlclient_r15
Ubuntu or Debian	\$ sudo apt-get install mysql-server

8. Change the `/etc/my.cnf` file as follows:

```

[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
bind-address=<ip-address>
default-storage-engine=InnoDB
sql_mode=STRICT_ALL_TABLES

```

9. Start the `mysql` daemon.

OS	Command
RHEL	\$ sudo service mysqld start

OS	Command
SLES and Ubuntu or Debian	\$ sudo service mysql start

- 10** Configure MySQL to use a strong password. In the following procedure, your current `root` password is blank. Press the **Enter** key when you're prompted for the 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!
```

- 11** Configure MySQL to start at boot.

OS	Command
RHEL	\$ 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
SLES	\$ sudo chkconfig --add mysql
Ubuntu or Debian	\$ sudo chkconfig mysql on

- 12** Create the Hue database and grant privileges to a `hue` user to manage the database.

```
mysql> create database hue;
Query OK, 1 row affected (0.01 sec)
mysql> grant all on hue.* to 'hue'@'localhost' identified by '<secretpassword>';
Query OK, 0 rows affected (0.00 sec)
```

- 13** Open the Hue configuration file in a text editor.

- 14** Edit the Hue configuration file `hue.ini`. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your setup):

```
host=localhost
port=3306
engine=mysql
user=hue
password=<secretpassword>
name=hue
```

- 15** As the `hue` user, load the existing data and create the necessary database tables using `syncdb` and `migrate` commands. When running these commands, Hue will try to access a `logs` directory, located at `/opt/cloudera/parcels/CDH/lib/hue/logs`, which might be missing. If that is the case, first create the `logs` directory and give the `hue` user and group ownership of the directory.

```
$ sudo mkdir /opt/cloudera/parcels/CDH/lib/hue/logs
$ sudo chown hue:hue /opt/cloudera/parcels/CDH/lib/hue/logs
```

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
$ mysql -u hue -p <secretpassword>
mysql > SHOW CREATE TABLE auth_permission;
```

16 (InnoDB only) Drop the foreign key.

```
mysql > ALTER TABLE auth_permission DROP FOREIGN KEY content_type_id_refs_id_XXXXXX;
```

17. Delete the rows in the `django_content_type` table.

```
mysql > DELETE FROM hue.django_content_type;
```

18 Load the data.

```
$ <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

19 (InnoDB only) Add the foreign key.

```
$ mysql -u hue -p <secretpassword>
mysql > ALTER TABLE auth_permission ADD FOREIGN KEY (`content_type_id`) REFERENCES
`django_content_type` (`id`);
```

Configuring the Hue Server to Store Data in PostgreSQL



Warning: Hue requires PostgreSQL 8.4 or newer.

1. Shut down the Hue server if it is running.
2. Dump the existing database data to a text file. Note that using the `.json` extension is required.



Note: `HUE_HOME` is a reference to the location of your Hue installation. For package installs, this is usually `/usr/lib/hue`; for parcel installs, this is usually, `/opt/cloudera/parcels/<parcel version>/lib/hue/`.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field. Here are some examples of JSON objects that should be deleted.

```
{
  "pk": 1,
  "model": "useradmin.userprofile",
  "fields": {
    "creation_method": "HUE",
    "user": 1,
    "home_directory": "/user/alice"
  }
},
{
  "pk": 2,
  "model": "useradmin.userprofile",
  "fields": {
    "creation_method": "HUE",
    "user": 1100714,
    "home_directory": "/user/bob"
  }
},
.....
```

4. Install required packages.

OS	Command
RHEL	\$ sudo yum install postgresql-devel gcc python-devel
SLES	\$ sudo zypper install postgresql-devel gcc python-devel
Ubuntu or Debian	\$ sudo apt-get install postgresql-devel gcc python-devel

5. Install the module that provides the connector to PostgreSQL.

```
sudo -u hue <HUE_HOME>/build/env/bin/pip install setuptools
sudo -u hue <HUE_HOME>/build/env/bin/pip install psycopg2
```

6. Install the PostgreSQL server.

OS	Command
RHEL	\$ sudo yum install postgresql-server
SLES	\$ sudo zypper install postgresql-server
Ubuntu or Debian	\$ sudo apt-get install postgresql

7. Initialize the data directories:

```
$ service postgresql initdb
```

8. Configure client authentication.

a. Edit `/var/lib/pgsql/data/pg_hba.conf`.

b. Set the authentication methods for local to `trust` and for host to `password` and add the following line at the end.

```
host hue hue 0.0.0.0/0 md5
```

9. Start the PostgreSQL server.

```
$ su - postgres
# /usr/bin/postgres -D /var/lib/pgsql/data > logfile 2>&1 &
```

10. Configure PostgreSQL to listen on all network interfaces.

Edit `/var/lib/pgsql/data/postgresql.conf` and set `listen_addresses`:

```
listen_addresses = '0.0.0.0' # Listen on all addresses
```

11. Create the hue database and grant privileges to a hue user to manage the database.

```
# psql -U postgres
postgres=# create database hue;
postgres=# \c hue;
You are now connected to database 'hue'.
postgres=# create user hue with password '<secretpassword>';
postgres=# grant all privileges on database hue to hue;
postgres=# \q
```

12 Restart the PostgreSQL server.

```
$ sudo service postgresql restart
```

13 Verify connectivity.

```
psql -h localhost -U hue -d hue
Password for user hue: <secretpassword>
```

14 Configure the PostgreSQL server to start at boot.

OS	Command
RHEL	\$ sudo /sbin/chkconfig postgresql on \$ sudo /sbin/chkconfig --list postgresql postgresql 0:off 1:off 2:on 3:on 4:on 5:on 6:off
SLES	\$ sudo chkconfig --add postgresql
Ubuntu or Debian	\$ sudo chkconfig postgresql on

15 Open the Hue configuration file in a text editor.

16 Edit the Hue configuration file `hue.ini`. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your setup):

```
host=localhost
port=5432
engine=postgresql_psycopg2
user=hue
password=<secretpassword>
name=hue
```

17 As the `hue` user, configure Hue to load the existing data and create the necessary database tables. You will need to run both the `migrate` and `syncdb` commands. When running these commands, Hue will try to access a `logs` directory, located at `/opt/cloudera/parcels/CDH/lib/hue/logs`, which might be missing. If that is the case, first create the `logs` directory and give the `hue` user and group ownership of the directory.

```
$ sudo mkdir /opt/cloudera/parcels/CDH/lib/hue/logs
$ sudo chown hue:hue /opt/cloudera/parcels/CDH/lib/hue/logs
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
```

18 Determine the foreign key ID.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# \d auth_permission;
```

19 Drop the foreign key that you retrieved in the previous step.

```
postgres=# ALTER TABLE auth_permission DROP CONSTRAINT content_type_id_refs_id_<XXXXXX>;
```

20 Delete the rows in the `django_content_type` table.

```
postgres=# TRUNCATE django_content_type CASCADE;
```

21 Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

2 Add back the foreign key you dropped.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# ALTER TABLE auth_permission ADD CONSTRAINT content_type_id_refs_id_<XXXXXX>
FOREIGN KEY (content_type_id) REFERENCES django_content_type(id) DEFERRABLE INITIALLY
DEFERRED;
```

Configuring the Hue Server to Store Data in Oracle



Important: Configure the database for character set AL32UTF8 and national character set UTF8.

1. Ensure Python 2.6 or newer is installed on the server Hue is running on.
2. Download the Oracle client libraries at [Instant Client for Linux x86-64](#) Version 11.1.0.7.0, Basic and SDK (with headers) zip files to the same directory.
3. Unzip the zip files.
4. Set environment variables to reference the libraries.

```
$ export ORACLE_HOME=<download directory>
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME
```

5. Create a symbolic link for the shared object:

```
$ cd $ORACLE_HOME
$ ln -sf libclntsh.so.11.1 libclntsh.so
```

6. Get a data dump by executing:



Note: HUE_HOME is a reference to the location of your Hue installation. For package installs, this is usually `/usr/lib/hue`; for parcel installs, this is usually, `/opt/cloudera/parcels/<parcel version>/lib/hue/`.

```
$ <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json --indent 2
```

7. Edit the Hue configuration file `hue.ini`. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your setup):

```
host=localhost
port=1521
engine=oracle
user=hue
password=<secretpassword>
name=<SID of the Oracle database, for example, 'XE'>
```

To use the Oracle service name instead of the SID, use the following configuration instead:

```
port=0
engine=oracle
user=hue
password=password
name=oracle.example.com:1521/orcl.example.com
```

The directive `port=0` allows Hue to use a service name. The `name` string is the connect string, including hostname, port, and service name.

To add support for a multithreaded environment, set the `threaded` option to `true` under the `[desktop]>[[database]]` section.

```
options={'threaded':true}
```

8. Grant required permissions to the Hue user in Oracle:

```
grant alter any index to hue;
grant alter any table to hue;
grant alter database link to hue;
grant create any index to hue;
grant create any sequence to hue;
grant create database link to hue;
grant create session to hue;
grant create table to hue;
grant drop any sequence to hue;
grant select any dictionary to hue;
grant drop any table to hue;
grant create procedure to hue;
grant create trigger to hue;
```

9. As the `hue` user, configure Hue to load the existing data and create the necessary database tables. You will need to run both the `syncdb` and `migrate` commands. When running these commands, Hue will try to access a `logs` directory, located at `/opt/cloudera/parcels/CDH/lib/hue/logs`, which might be missing. If that is the case, first create the `logs` directory and give the `hue` user and group ownership of the directory.

```
$ sudo mkdir /opt/cloudera/parcels/CDH/lib/hue/logs
$ sudo chown hue:hue /opt/cloudera/parcels/CDH/lib/hue/logs
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
```

10. Ensure you are connected to Oracle as the `hue` user, then run the following command to delete all data from Oracle tables:

```
SELECT 'DELETE FROM ' || '.' || table_name || ';' FROM user_tables;
```

11. Run the statements generated in the preceding step.

12. Commit your changes.

```
commit;
```

13. Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

Managing Impala

This section explains how to configure Impala to accept connections from applications that use popular programming APIs:

- [Post-Installation Configuration for Impala](#) on page 187
- [Configuring Impala to Work with ODBC](#) on page 189
- [Configuring Impala to Work with JDBC](#) on page 191

This type of configuration is especially useful when using Impala in combination with Business Intelligence tools, which use these standard interfaces to query different kinds of database and Big Data systems.

You can also configure these other aspects of Impala:

- [Overview of Impala Security](#)
- [Modifying Impala Startup Options](#)

The Impala Service

The Impala Service is the Cloudera Manager representation of the three daemons that make up the Impala interactive SQL engine. Through the Impala Service page, you can monitor, start and stop, and configure all the related daemons from a central page.

For general information about Impala and how to use it, especially for writing Impala SQL queries, see [Cloudera Impala Guide](#).

For information on features that support Impala resource management see [Admission Control and Query Queuing](#) on page 254.

Installing Impala and Creating the Service

You can install Impala through the Cloudera Manager installation wizard, using either parcels or packages, and have the service created and started as part of the Installation wizard. See [Installing Impala](#).

If you elect not to include the Impala service using the Installation wizard, you can use the **Add Service** wizard to perform the installation. The wizard will automatically configure and start the dependent services and the Impala service. See [Adding a Service](#) on page 36 for instructions.

For general information about Impala and how to use it, see [Cloudera Impala Guide](#).

For information on features that support Impala resource management see [Impala Resource Management](#) on page 254.

Configuring the Impala Service

There are several types of configuration settings you may need to apply, depending on your situation.

Configuring Table Statistics

Configuring table statistics is highly recommended when using Impala. It allows Impala to make optimizations that can result in significant (over 10x) performance improvement for some joins. If these are not available, Impala will still function, but at lower performance.

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, follow the procedure in [Hive Table Statistics](#) on page 156.

Using a Load Balancer with Impala

To configure a load balancer:

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Scope > Impala Daemon**
4. Select **Category > All**
5. Enter the hostname and port number of the load balancer in the **Impala Daemons Load Balancer** property in the format `hostname:port number`.



Note:

When you set this property, Cloudera Manager regenerates the keytabs for Impala Daemon roles. The principal in these keytabs contains the load balancer hostname.

If there is a Hue service that depends on this Impala service, it also uses the load balancer to communicate with Impala.

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

Impala Web Servers

Enabling and Disabling Access to Impala Web Servers

Each of the Impala-related daemons includes a built-in web server that lets an administrator diagnose issues with each daemon on a particular host, or perform other administrative actions such as cancelling a running query. By default, these web servers are enabled. You might turn them off in a high-security configuration where it is not appropriate for users to have access to this kind of monitoring information through a web interface. (To leave the web servers enabled but control who can access their web pages, consult the *Configuring Secure Access for Impala Web Servers* later in this section.)

- **Impala Daemon**
 1. Go to the Impala service.
 2. Click the **Configuration** tab.
 3. Select **Scope > Impala Daemon**
 4. Select **Category > Ports and Addresses**.
 5. Select or deselect **Enable Impala Daemon Web Server**.
 6. Click **Save Changes** to commit the changes.
 7. Restart the Impala service.
- **Impala StateStore**
 1. Go to the Impala service.
 2. Click the **Configuration** tab.
 3. Select **Scope > Impala StateStore**.
 4. Select **Category > All**
 5. Select or deselect **Enable StateStore Web Server**.
 6. Click **Save Changes** to commit the changes.
 7. Restart the Impala service.
- **Impala Catalog Server**
 1. Go to the Impala service.
 2. Click the **Configuration** tab.
 3. Select **Scope > Impala Catalog Server**.
 4. Select **Category > All**
 5. Check or uncheck **Enable Catalog Server Web Server**.
 6. Click **Save Changes** to commit the changes.
 7. Restart the Impala service.

Opening Impala Web Server UIs

- **Impala StateStore**
 1. Go to the Impala service.
 2. Select **Web UI > Impala StateStore Web UI**.
- **Impala Daemon**
 1. Go to the Impala service.
 2. Click the **Instances** tab.
 3. Click an **Impala Daemon** instance.
 4. Click **Impala Daemon Web UI**.
- **Impala Catalog Server**
 1. Go to the Impala service.

2. Select **Web UI > Impala Catalog Web UI**.

- **Impala Llama ApplicationMaster**

1. Go to the Impala service.
2. Click the **Instances** tab.
3. Click a **Impala Llama ApplicationMaster** instance.
4. Click **Llama Web UI**.

Configuring Secure Access for Impala Web Servers

Cloudera Manager supports two methods of authentication for secure access to the Impala Catalog Server, Daemon, and StateStore web servers: password-based authentication and SSL certificate authentication. Both of these can be configured through properties of the Impala Catalog Server, Daemon, and StateStore. Authentication for the three types of daemons can be configured independently.

Configuring Password Authentication

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Search for "password" using the Search box within the Configuration page. This should display the password-related properties (Username and Password properties) for the Impala Catalog Server, Daemon, and StateStore. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
4. Enter a username and password into these fields.
5. Click **Save Changes** to commit the changes.
6. Restart the Impala service.

Now when you access the Web UI for the Impala Catalog Server, Daemon, and StateStore, you are asked to log in before access is granted.

Configuring SSL Certificate Authentication

1. Create or obtain an SSL certificate.
2. Place the certificate, in `.pem` format, on the hosts where the Impala Catalog Server and StateStore are running, and on each host where an Impala Daemon is running. It can be placed in any location (path) you choose. If all the Impala Daemons are members of the same role group, then the `.pem` file must have the same path on every host.
3. Go to the Impala service page.
4. Click the **Configuration** tab.
5. Search for "certificate" using the Search box within the Configuration page. This should display the certificate file location properties for the Impala Catalog Server, Daemon, and StateStore. If there are multiple role groups configured for Impala Daemon instances, the search should display all of them.
6. In the property fields, enter the full path name to the certificate file.
7. Click **Save Changes** to commit the changes.
8. Restart the Impala service.



Important: If Cloudera Manager cannot find the `.pem` file on the host for a specific role instance, that role will fail to start.

When you access the Web UI for the Impala Catalog Server, Daemon, and StateStore, `https` will be used.

Post-Installation Configuration for Impala

This section describes the mandatory and recommended configuration settings for Cloudera Impala. If Impala is installed using Cloudera Manager, some of these configurations are completed automatically; you must still configure short-circuit

reads manually. If you installed Impala without Cloudera Manager, or if you want to customize your environment, consider making the changes described in this topic.

In some cases, depending on the level of Impala, CDH, and Cloudera Manager, you might need to add particular component configuration details in one of the free-form fields on the Impala configuration pages within Cloudera Manager. In Cloudera Manager 4, these fields are labelled **Safety Valve**; in Cloudera Manager 5, they are called **Advanced Configuration Snippet**.

- You must enable short-circuit reads, whether or not Impala was installed through Cloudera Manager. This setting goes in the Impala configuration settings, not the Hadoop-wide settings.
- If you installed Impala in an environment that is not managed by Cloudera Manager, you must enable block location tracking, and you can optionally enable native checksumming for optimal performance.
- If you deployed Impala using Cloudera Manager see [Testing Impala Performance](#) to confirm proper configuration.

Mandatory: Short-Circuit Reads

Enabling short-circuit reads allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes, improving performance. This setting also minimizes the number of additional copies of data. Short-circuit reads requires `libhadoop.so` (the Hadoop Native Library) to be accessible to both the server and the client. `libhadoop.so` is not available if you have installed from a tarball. You must install from an `.rpm`, `.deb`, or `parcel` to use short-circuit local reads.



Note: If you use Cloudera Manager, you can enable short-circuit reads through a checkbox in the user interface and that setting takes effect for Impala as well.

To configure DataNodes for short-circuit reads:

1. Copy the client `core-site.xml` and `hdfs-site.xml` configuration files from the Hadoop configuration directory to the Impala configuration directory. The default Impala configuration location is `/etc/impala/conf`.
2. On all Impala nodes, configure the following properties in Impala's copy of `hdfs-site.xml` as shown:

```
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>true</value>
</property>

<property>
  <name>dfs.domain.socket.path</name>
  <value>/var/run/hdfs-sockets/dn</value>
</property>

<property>
  <name>dfs.client.file-block-storage-locations.timeout.millis</name>
  <value>10000</value>
</property>
```

3. If `/var/run/hadoop-hdfs/` is group-writable, make sure its group is `root`.



Note: If you are also going to enable block location tracking, you can skip copying configuration files and restarting DataNodes and go straight to [Optional: Block Location Tracking](#). Configuring short-circuit reads and block location tracking require the same process of copying files and restarting services, so you can complete that process once when you have completed all configuration changes. Whether you copy files and restart services now or during configuring block location tracking, short-circuit reads are not enabled until you complete those final steps.

4. After applying these changes, restart all DataNodes.

Mandatory: Block Location Tracking

Enabling block location metadata allows Impala to know which disk data blocks are located on, allowing better utilization of the underlying disks. Impala will not start unless this setting is enabled.

To enable block location tracking:

1. For each DataNode, adding the following to the `hdfs-site.xml` file:

```
<property>
  <name>dfs.datanode.hdfs-blocks-metadata.enabled</name>
  <value>true</value>
</property>
```

2. Copy the client `core-site.xml` and `hdfs-site.xml` configuration files from the Hadoop configuration directory to the Impala configuration directory. The default Impala configuration location is `/etc/impala/conf`.
3. After applying these changes, restart all DataNodes.

Optional: Native Checksumming

Enabling native checksumming causes Impala to use an optimized native library for computing checksums, if that library is available.

To enable native checksumming:

If you installed CDH from packages, the native checksumming library is installed and setup correctly. In such a case, no additional steps are required. Conversely, if you installed by other means, such as with tarballs, native checksumming may not be available due to missing shared objects. Finding the message "Unable to load native-hadoop library for your platform... using builtin-java classes where applicable" in the Impala logs indicates native checksumming may be unavailable. To enable native checksumming, you must build and install `libhadoop.so` (the [Hadoop Native Library](#)).

Configuring Impala to Work with ODBC

Third-party products can be designed to integrate with Impala using ODBC. For the best experience, ensure any third-party product you intend to use is supported. Verifying support includes checking that the versions of Impala, ODBC, the operating system, and the third-party product have all been approved for use together. Before configuring your systems to use ODBC, download a connector. You may need to sign in and accept license agreements before accessing the pages required for downloading ODBC connectors.

Downloading the ODBC Driver



Important: As of late 2015, most business intelligence applications are certified with the 2.x ODBC drivers. Although the instructions on this page cover both the 2.x and 1.x drivers, expect to use the 2.x drivers exclusively for most ODBC applications connecting to Impala.

See the [documentation page](#) for installation instructions.

Configuring the ODBC Port

Versions 2.5 and 2.0 of the Cloudera ODBC Connector, currently certified for some but not all BI applications, use the HiveServer2 protocol, corresponding to Impala port 21050. Impala supports Kerberos authentication with all the supported versions of the driver, and requires ODBC 2.05.13 for Impala or higher for LDAP username/password authentication.

Version 1.x of the Cloudera ODBC Connector uses the original HiveServer1 protocol, corresponding to Impala port 21000.

Example of Setting Up an ODBC Application for Impala

To illustrate the outline of the setup process, here is a transcript of a session to set up all required drivers and a business intelligence application that uses the ODBC driver, under Mac OS X. Each `.dmg` file runs a GUI-based installer, first for

the [underlying ODBC driver](#) needed for non-Windows systems, then for the Cloudera ODBC Connector, and finally for the BI tool itself.

```

$ ls -l
Cloudera-ODBC-Driver-for-Impala-Install-Guide.pdf
BI_Tool_Installer.dmg
iodbc-sdk-3.52.7-macosx-10.5.dmg
ClouderaImpalaODBC.dmg
$ open iodbc-sdk-3.52.7-macosx-10.dmg
Install the ODBC driver using its installer
$ open ClouderaImpalaODBC.dmg
Install the Cloudera ODBC Connector using its installer
$ installer_dir=$(pwd)
$ cd /opt/cloudera/impalaodbc
$ ls -l
Cloudera ODBC Driver for Impala Install Guide.pdf
Readme.txt
Setup
lib
ErrorMessages
Release Notes.txt
Tools
$ cd Setup
$ ls
odbc.ini      odbcinst.ini
$ cp odbc.ini ~/.odbc.ini
$ vi ~/.odbc.ini
$ cat ~/.odbc.ini
[ODBC]
# Specify any global ODBC configuration here such as ODBC tracing.

[ODBC Data Sources]
Sample Cloudera Impala DSN=Cloudera ODBC Driver for Impala

[Sample Cloudera Impala DSN]

# Description: DSN Description.
# This key is not necessary and is only to give a description of the data source.
Description=Cloudera ODBC Driver for Impala DSN

# Driver: The location where the ODBC driver is installed to.
Driver=/opt/cloudera/impalaodbc/lib/universal/libclouderaimpalaodbc.dylib

# The DriverUnicodeEncoding setting is only used for SimbaDM
# When set to 1, SimbaDM runs in UTF-16 mode.
# When set to 2, SimbaDM runs in UTF-8 mode.
#DriverUnicodeEncoding=2

# Values for HOST, PORT, KrbFQDN, and KrbServiceName should be set here.
# They can also be specified on the connection string.
HOST=hostname.sample.example.com
PORT=21050
Schema=default

# The authentication mechanism.
# 0 - No authentication (NOSASL)
# 1 - Kerberos authentication (SASL)
# 2 - Username authentication (SASL)
# 3 - Username/password authentication (SASL)
# 4 - Username/password authentication with SSL (SASL)
# 5 - No authentication with SSL (NOSASL)
# 6 - Username/password authentication (NOSASL)
AuthMech=0

# Kerberos related settings.
KrbFQDN=
KrbRealm=
KrbServiceName=

# Username/password authentication with SSL settings.
UID=
PWD=
CAIssuedCertNamesMismatch=1

```

```
TrustedCerts=/opt/cloudera/impalaodbc/lib/universal/cacerts.pem

# Specify the proxy user ID to use.
#DelegationUID=

# General settings
TSaslTransportBufSize=1000
RowsFetchedPerBlock=10000
SocketTimeout=0
StringColumnLength=32767
UseNativeQuery=0
$ pwd
/opt/cloudera/impalaodbc/Setup
$ cd $installer_dir
$ open BI_Tool_Installer.dmg
Install the BI tool using its installer
$ ls /Applications | grep BI_Tool
BI_Tool.app
$ open -a BI_Tool.app
In the BI tool, connect to a data source using port 21050
```

Notes about JDBC and ODBC Interaction with Impala SQL Features

Most Impala SQL features work equivalently through the `impala-shell` interpreter of the JDBC or ODBC APIs. The following are some exceptions to keep in mind when switching between the interactive shell and applications using the APIs:



Note: If your JDBC or ODBC application connects to Impala through a load balancer such as `haproxy`, be cautious about reusing the connections. If the load balancer has set up connection timeout values, either check the connection frequently so that it never sits idle longer than the load balancer timeout value, or check the connection validity before using it and create a new one if the connection has been closed.

Configuring Impala to Work with JDBC

Impala supports the standard JDBC interface, allowing access from commercial Business Intelligence tools and custom software written in Java or other programming languages. The JDBC driver allows you to access Impala from a Java program that you write, or a Business Intelligence or similar tool that uses JDBC to communicate with various database products.

Setting up a JDBC connection to Impala involves the following steps:

- Verifying the communication port where the Impala daemons in your cluster are listening for incoming JDBC requests.
- Installing the JDBC driver on every system that runs the JDBC-enabled application.
- Specifying a connection string for the JDBC application to access one of the servers running the `impalad` daemon, with the appropriate security settings.

Configuring the JDBC Port

The default port used by JDBC 2.0 and later (as well as ODBC 2.x) is 21050. Impala server accepts JDBC connections through this same port 21050 by default. Make sure this port is available for communication with other hosts on your network, for example, that it is not blocked by firewall software. If your JDBC client software connects to a different port, specify that alternative port number with the `--hs2_port` option when starting `impalad`. See [Starting Impala](#) for details about Impala startup options. See [Ports Used by Impala](#) for information about all ports used for communication between Impala and clients or between Impala components.

Choosing the JDBC Driver

In Impala 2.0 and later, you have the choice between the Cloudera JDBC Connector and the Hive 0.13 JDBC driver. Cloudera recommends using the Cloudera JDBC Connector where practical.

If you are already using JDBC applications with an earlier Impala release, you must update your JDBC driver to one of these choices, because the Hive 0.12 driver that was formerly the only choice is not compatible with Impala 2.0 and later.

Both the Cloudera JDBC 2.5 Connector and the Hive JDBC driver provide a substantial speed increase for JDBC applications with Impala 2.0 and higher, for queries that return large result sets.

Enabling Impala JDBC Support on Client Systems

Using the Cloudera JDBC Connector (recommended)

You download and install the Cloudera JDBC 2.5 connector on any Linux, Windows, or Mac system where you intend to run JDBC-enabled applications. From the [Cloudera Connectors download page](#), you choose the appropriate protocol (JDBC or ODBC) and target product (Impala or Hive). The ease of downloading and installing on non-CDH systems makes this connector a convenient choice for organizations with heterogeneous environments.

Using the Hive JDBC Driver

You install the Hive JDBC driver (`hive-jdbc` package) through the Linux package manager, on hosts within the CDH cluster. The driver consists of several Java JAR files. The same driver can be used by Impala and Hive.

To get the JAR files, install the Hive JDBC driver on each CDH-enabled host in the cluster that will run JDBC applications. Follow the instructions for [CDH 5](#).



Note: The latest JDBC driver, corresponding to Hive 0.13, provides substantial performance improvements for Impala queries that return large result sets. Impala 2.0 and later are compatible with the Hive 0.13 driver. If you already have an older JDBC driver installed, and are running Impala 2.0 or higher, consider upgrading to the latest Hive JDBC driver for best performance with JDBC applications.

If you are using JDBC-enabled applications on hosts outside the CDH cluster, you cannot use the CDH install procedure on the non-CDH hosts. Install the JDBC driver on at least one CDH host using the preceding procedure. Then download the JAR files to each client machine that will use JDBC with Impala:

```
commons-logging-X.X.X.jar
hadoop-common.jar
hive-common-X.XX.X-cdhX.X.X.jar
hive-jdbc-X.XX.X-cdhX.X.X.jar
hive-metastore-X.XX.X-cdhX.X.X.jar
hive-service-X.XX.X-cdhX.X.X.jar
httpclient-X.X.X.jar
httpcore-X.X.X.jar
libfb303-X.X.X.jar
libthrift-X.X.X.jar
log4j-X.X.XX.jar
slf4j-api-X.X.X.jar
slf4j-logXjXX-X.X.X.jar
```

To enable JDBC support for Impala on the system where you run the JDBC application:

1. Download the JAR files listed above to each client machine.



Note: For Maven users, see [this sample github page](#) for an example of the dependencies you could add to a `pom` file instead of downloading the individual JARs.

2. Store the JAR files in a location of your choosing, ideally a directory already referenced in your `CLASSPATH` setting. For example:
 - On Linux, you might use a location such as `/opt/jars/`.
 - On Windows, you might use a subdirectory underneath `C:\Program Files`.

3. To successfully load the Impala JDBC driver, client programs must be able to locate the associated JAR files. This often means setting the `CLASSPATH` for the client process to include the JARs. Consult the documentation for your JDBC client for more details on how to install new JDBC drivers, but some examples of how to set `CLASSPATH` variables include:

- On Linux, if you extracted the JARs to `/opt/jars/`, you might issue the following command to prepend the JAR files path to an existing classpath:

```
export CLASSPATH=/opt/jars/*.jar:$CLASSPATH
```

- On Windows, use the **System Properties** control panel item to modify the **Environment Variables** for your system. Modify the environment variables to include the path to which you extracted the files.



Note: If the existing `CLASSPATH` on your client machine refers to some older version of the Hive JARs, ensure that the new JARs are the first ones listed. Either put the new JAR files earlier in the listings, or delete the other references to Hive JAR files.

Establishing JDBC Connections

The JDBC driver class depends on which driver you select.



Note: If your JDBC or ODBC application connects to Impala through a load balancer such as `haproxy`, be cautious about reusing the connections. If the load balancer has set up connection timeout values, either check the connection frequently so that it never sits idle longer than the load balancer timeout value, or check the connection validity before using it and create a new one if the connection has been closed.

Using the Cloudera JDBC Connector (recommended)

Depending on the level of the JDBC API your application is targeting, you can use the following fully-qualified class names (FQCNs):

- `com.cloudera.impala.jdbc41.Driver`
- `com.cloudera.impala.jdbc41.DataSource`
- `com.cloudera.impala.jdbc4.Driver`
- `com.cloudera.impala.jdbc4.DataSource`
- `com.cloudera.impala.jdbc3.Driver`
- `com.cloudera.impala.jdbc3.DataSource`

The connection string has the following format:

```
jdbc:impala://Host:Port[/Schema];Property1=Value;Property2=Value;...
```

The `port` value is typically 21050 for Impala.

For full details about the classes and the connection string (especially the property values available for the connection string), download the appropriate driver documentation for your platform from [the Impala JDBC Connector download page](#).

Using the Hive JDBC Driver

For example, with the Hive JDBC driver, the class name is `org.apache.hive.jdbc.HiveDriver`. Once you have configured Impala to work with JDBC, you can establish connections between the two. To do so for a cluster that does

Managing CDH and Managed Services

not use Kerberos authentication, use a connection string of the form `jdbc:hive2://host:port/;auth=noSasl`. For example, you might use:

```
jdbc:hive2://myhost.example.com:21050/;auth=noSasl
```

To connect to an instance of Impala that requires Kerberos authentication, use a connection string of the form `jdbc:hive2://host:port/;principal=principal_name`. The principal must be the same user principal you used when starting Impala. For example, you might use:

```
jdbc:hive2://myhost.example.com:21050/;principal=impala/myhost.example.com@H2.EXAMPLE.COM
```

To connect to an instance of Impala that requires LDAP authentication, use a connection string of the form `jdbc:hive2://host:port/db_name;user=ldap_userid;password=ldap_password`. For example, you might use:

```
jdbc:hive2://myhost.example.com:21050/test_db;user=fred;password=xyz123
```



Note:

Currently, the Hive JDBC driver does not support connections that use both Kerberos authentication and SSL encryption. To use both of these security features with Impala through a JDBC application, use the [Cloudera JDBC Connector](#) as the JDBC driver.

Managing Isilon

EMC Isilon is a storage service with a distributed file system that can be used in place of HDFS to provide storage for CDH services.



Note: This documentation covers only the Cloudera Manager portion of using EMC Isilon storage with Cloudera Manager. For information about tasks performed on Isilon OneFS, see the information hub for Cloudera on the EMC Community Network: <https://community.emc.com/docs/DOC-39529>.

Supported Versions

The following versions of Cloudera and Isilon products are supported:

- CDH 5.4.4 or higher, and corresponding level of Impala
- Cloudera Manager 5.4.3 or higher
- Isilon OneFS 7.2.0.3



Note: Cloudera Navigator is not supported in this release.

Differences between Isilon HDFS and CDH HDFS

The following features of HDFS are not implemented with Isilon OneFS:

- HDFS caching
- HDFS encryption
- HDFS ACLs

Preliminary Steps on the Isilon Service

Before installing a Cloudera Manager cluster to use Isilon storage, perform the following steps on the Isilon OneFS system. For detailed information on setting up Isilon OneFS for Cloudera Manager, see the Isilon documentation at <https://community.emc.com/docs/DOC-39529>.

1. Create an Isilon access zone with HDFS support.

Example:
`/ifs/your-access-zone/hdfs`



Note: The above is simply an example; the HDFS root directory does not have to begin with `ifs` or end with `hdfs`.

2. Create two directories that will be used by all CDH services:

a. Create a `tmp` directory in the access zone.

- Create `supergroup` group and `hdfs` user.
- Create a `tmp` directory and set ownership to `hdfs:supergroup`, and permissions to `1777`.

Example:
`cd hdfs_root_directory`
`isi_run -z zone_id mkdir tmp`
`isi_run -z zone_id chown hdfs:supergroup tmp`
`isi_run -z zone_id chmod 1777 tmp`

b. Create a `user` directory in the access zone and set ownership to `hdfs:supergroup`, and permissions to `755`

Example:
`cd hdfs_root_directory`
`isi_run -z zone_id mkdir user`
`isi_run -z zone_id chown hdfs:supergroup user`
`isi_run -z zone_id chmod 755 user`

3. Create the service-specific users, groups, or directories for each CDH service you plan to use. Create the directories under the access zone you have created.



Note: Many of the values provided in the examples below are default values in Cloudera Manager and must match the Cloudera Manager configuration settings. The format for the examples is: `dir user:group permission`. Create the directories below under the access zone you have created, for example, `/ifs/your-access-zone/hdfs/`

- ZooKeeper: nothing required.
- HBase
 - Create `hbase` group with `hbase` user.
 - Create the root directory for HBase:

Example:
`hdfs_root_directory/hbase hbase:hbase 755`

- YARN (MR2)
 - Create `mapred` group with `mapred` user.
 - Create `history` directory for YARN:

Example:
`hdfs_root_directory/user/history mapred:hadoop 777`

- Create the remote application log directory for YARN:

```
Example:  
hdfs_root_directory/tmp/logs mapred:hadoop 775
```

- Oozie
 - Create oozie group with oozie user.
 - Create the user directory for Oozie:

```
Example:  
hdfs_root_directory/user/oozie oozie:oozie 775
```

- Flume
 - Create flume group with flume user.
 - Create the user directory for Flume:

```
Example:  
hdfs_root_directory/user/flume flume:flume 775
```

- Hive
 - Create hive group with hive user.
 - Create the user directory for Hive:

```
Example:  
hdfs_root_directory/user/hive hive:hive 775
```

- Create the warehouse directory for Hive:

```
Example:  
hdfs_root_directory/user/hive/warehouse hive:hive 1777
```

- Create a temporary directory for Hive:

```
Example:  
hdfs_root_directory/tmp/hive hive:supergroup 777
```

- Solr
 - Create solr group with solr user.
 - Create the data directory for Solr:

```
Example:  
hdfs_root_directory/solr solr:solr 775
```

- Sqoop
 - Create sqoop group with sqoop2 user.
 - Create the user directory for Sqoop:

```
Example:  
hdfs_root_directory/user/sqoop2 sqoop2:sqoop 775
```

- Hue
 - Create hue group with hue user.

- Create `sample` group with `sample` user.
- Spark
 - Create `spark` group with `spark` user.
 - Create the user directory for Spark:

```
Example:
hdfs_root_directory/user/spark spark:spark 751
```

- Create application history directory for Spark:

```
Example:
hdfs_root_directory/user/spark/applicationHistory spark:spark 1777
```

Once the users, groups, and directories are created in Isilon OneFS, you are ready to install Cloudera Manager.

Installing Cloudera Manager with Isilon

To install Cloudera Manager following the instructions provided in [Installing Cloudera Manager, CDH, and Managed Services](#).

- The simplest installation procedure, suitable for development or proof of concept, is [Installation Path A](#), which uses embedded databases that are installed as part of the Cloudera Manager installation process.
- For production environments, [Installation Path B - Manual Installation Using Cloudera Manager Packages](#) describes configuring external databases for Cloudera Manager and CDH storage needs.

If you choose parcel installation on the **Cluster Installation** screen, the installation wizard will point to the latest parcels of CDH available.

On the installation wizard's **Cluster Setup** page, choose **Custom Services**, and choose the services you want installed in the cluster. Be sure to choose **Isilon** among the selected services, do not select the **HDFS** service, and do not check **Include Cloudera Navigator** at the bottom of the **Cluster Setup** page. Also, on the **Role Assignments** page, be sure to specify the hosts that will serve as gateway roles for the Isilon service. You can add gateway roles to one, some, or all nodes in the cluster.

Installing a Secure Cluster with Isilon

To set up a secure cluster with Isilon using Kerberos, perform the following steps:

1. Create an unsecure Cloudera Manager cluster as described above in [Installing Cloudera Manager with Isilon](#) on page 197.
2. Follow the Isilon documentation to enable Kerberos for your access zone: <https://community.emc.com/docs/DOC-39529>. This includes adding a Kerberos authentication provider to your Isilon access zone.
3. Add the following proxy users in Isilon if your Cloudera Manager cluster includes the corresponding CDH services. The procedure for configuring proxy users is described in the Isilon documentation, <https://community.emc.com/docs/DOC-39529>.
 - proxy user `hdfs` for `hdfs` user.
 - proxy user `mapred` for `mapred` user.
 - proxy user `hive` for `hive` user.
 - proxy user `impala` for `impala` user.
 - proxy user `oozie` for `oozie` user
 - proxy user `flume` for `flume` user
 - proxy user `hue` for `hue` user
4. Follow the Cloudera Manager documentation for information on configuring a secure cluster with Kerberos: [Configuring Authentication in Cloudera Manager](#).

Using Impala with Isilon Storage

You can use Impala to query data files that reside on EMC Isilon storage devices, rather than in HDFS. This capability allows convenient query access to a storage system where you might already be managing large volumes of data. The combination of the Impala query engine and Isilon storage is certified on CDH 5.4.4 through CDH 5.15.

Because the EMC Isilon storage devices use a global value for the block size rather than a configurable value for each file, the `PARQUET_FILE_SIZE` query option has no effect when Impala inserts data into a table or partition residing on Isilon storage. Use the `isi` command to set the default block size globally on the Isilon device. For example, to set the Isilon default block size to 256 MB, the recommended size for Parquet data files for Impala, issue the following command:

```
isi hdfs settings modify --default-block-size=256MB
```

The typical use case for Impala and Isilon together is to use Isilon for the default filesystem, replacing HDFS entirely. In this configuration, when you create a database, table, or partition, the data always resides on Isilon storage and you do not need to specify any special `LOCATION` attribute. If you do specify a `LOCATION` attribute, its value refers to a path within the Isilon filesystem. For example:

```
-- If the default filesystem is Isilon, all Impala data resides there
-- and all Impala databases and tables are located there.
CREATE TABLE t1 (x INT, s STRING);

-- You can specify LOCATION for database, table, or partition,
-- using values from the Isilon filesystem.
CREATE DATABASE d1 LOCATION '/some/path/on/isilon/server/d1.db';
CREATE TABLE d1.t2 (a TINYINT, b BOOLEAN);
```

Impala can write to, delete, and rename data files and database, table, and partition directories on Isilon storage. Therefore, Impala statements such as `CREATE TABLE`, `DROP TABLE`, `CREATE DATABASE`, `DROP DATABASE`, `ALTER TABLE`, and `INSERT` work the same with Isilon storage as with HDFS.

When the Impala spill-to-disk feature is activated by a query that approaches the memory limit, Impala writes all the temporary data to a local (not Isilon) storage device. Because the I/O bandwidth for the temporary data depends on the number of local disks, and clusters using Isilon storage might not have as many local disks attached, pay special attention on Isilon-enabled clusters to any queries that use the spill-to-disk feature. Where practical, tune the queries or allocate extra memory for Impala to avoid spilling. Although you can specify an Isilon storage device as the destination for the temporary data for the spill-to-disk feature, that configuration is not recommended due to the need to transfer the data both ways using remote I/O.

When tuning Impala queries on HDFS, you typically try to avoid any remote reads. When the data resides on Isilon storage, all the I/O consists of remote reads. Do not be alarmed when you see non-zero numbers for remote read measurements in query profile output. The benefit of the Impala and Isilon integration is primarily convenience of not having to move or copy large volumes of data to HDFS, rather than raw query performance. You can increase the performance of Impala I/O for Isilon systems by increasing the value for the `num_remote_hdfs_io_threads` configuration parameter, in the Cloudera Manager user interface for clusters using Cloudera Manager, or through the `--num_remote_hdfs_io_threads` startup option for the `impalad` daemon on clusters not using Cloudera Manager.

For information about managing Isilon storage devices through Cloudera Manager, see [Managing Isilon](#) on page 194.

Required Configurations

Specify the following configurations in Cloudera Manager on the **Clusters > Isilon Service > Configuration** tab:

- In **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** `hdfs-site.xml` and the **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** properties for the Isilon service, set the value of the `dfs.client.file-block-storage-locations.timeout.millis` property to 10000.
- In the Isilon **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** property for the Isilon service, set the value of the `hadoop.security.token.service.use_ip` property to `FALSE`.
- If you see errors that reference the `.Trash` directory, make sure that the **Use Trash** property is selected.

Managing Key-Value Store Indexer

The Key-Value Store Indexer service uses the [Lily HBase Indexer Service](#) to index the stream of records being added to HBase tables. Indexing allows you to query data stored in HBase with the [Solr service](#).

The Key-Value Store Indexer service is installed in the same parcel or package along with the CDH 5 or Solr service.

Adding the Key-Value Store Indexer Service

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

1. On the **Home** > **Status** tab, click



to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.

2. Select the **Key-Value Store Indexer** service and click **Continue**.
3. Select the radio button next to the services on which the new service should depend. All services must depend on the *same* ZooKeeper service. Click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. Click **Continue**.
6. Review the configuration changes to be applied. Confirm the settings entered for file system paths. The file paths required vary based on the services to be installed. If you chose to add the Sqoop service, indicate whether to use the default Derby database or the embedded PostgreSQL database. If the latter, type the database name, host, and user credentials that you specified when you created the database.



Warning: Do not place DataNode data directories on NAS devices. When resizing an NAS, block replicas can be deleted, which will result in reports of missing blocks.

Click **Continue**. The wizard starts the services.

7. Click **Continue**.
8. Click **Finish**.

Enabling Morphlines with Search and HBase Indexing

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

Cloudera Morphlines is an open source framework that reduces the time and skills necessary to build or change Search indexing applications. A morphline is a rich configuration file that simplifies defining an ETL transformation chain.

1. Go to the Indexer service.
2. Click the **Configuration** tab.
3. Select **Scope > All**.
4. Select **Category > Morphlines**.
5. Create the necessary configuration files, and modify the content in the following properties:
 - **Morphlines File** — Text that goes into the `morphlines.conf` used by HBase indexers. You should use `$ZK_HOST` in this file instead of specifying a ZooKeeper quorum. Cloudera Manager automatically replaces the `$ZK_HOST` variable with the correct value during the Solr configuration deployment.
 - **Custom MIME-types File** — Text that goes verbatim into the `custom-mimetypes.xml` file used by HBase Indexers with the `detectMimeTypes` command. See the [Cloudera Morphlines Reference Guide](#) for details on this command.
 - **Grok Dictionary File** — Text that goes verbatim into the `grok-dictionary.conf` file used by HBase Indexers with the `grok` command. See the [Cloudera Morphlines Reference Guide](#) for details of this command.

See [Extracting, Transforming, and Loading Data With Cloudera Morphlines](#) for information about using morphlines with Search and HBase.

Managing MapReduce and YARN

CDH supports two versions of the MapReduce computation framework: MRv1 and MRv2, which are implemented by the [MapReduce](#) (MRv1) and [YARN](#) (MRv2) services. YARN is backwards-compatible with MapReduce. (All jobs that run against MapReduce will also run in a YARN cluster).

The MRv2 YARN architecture splits the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM) form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on worker hosts instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework-specific library and negotiates resources from the ResourceManager and works with the NodeManagers to execute and monitor the tasks. For details of this architecture, see [Apache Hadoop NextGen MapReduce \(YARN\)](#).

- The Cloudera Manager Admin Console has different methods for displaying MapReduce and YARN job history. See [Monitoring MapReduce Jobs](#) and [Monitoring YARN Applications](#).
- For information on configuring the MapReduce and YARN services for high availability, see [MapReduce \(MRv1\) and YARN \(MRv2\) High Availability](#) on page 309
- For information on configuring MapReduce and YARN resource management features, see [Resource Management](#) on page 236.

Defaults and Recommendations

- In a Cloudera Manager deployment of a CDH 4 cluster, the MapReduce service is the default MapReduce computation framework. You can create a YARN service in a CDH 4 cluster, but it is not considered production ready.
- In a Cloudera Manager deployment of a CDH 5 cluster, the YARN service is the default MapReduce computation framework. In CDH 5, the MapReduce service has been deprecated. However, the MapReduce service is fully supported for backward compatibility through the CDH 5 lifecycle.
- For production uses, Cloudera recommends that *only one* MapReduce framework should be running at any given time. If development needs or other use case requires switching between MapReduce and YARN, both services can be configured at the same time, but only one should be in a running (to fully optimize the hardware resources available).

Migrating from MapReduce to YARN

Cloudera Manager provides a wizard described in [Importing MapReduce Configurations to YARN](#) on page 206 to easily migrate MapReduce configurations to YARN. The wizard performs all the steps ([Switching Between MapReduce and YARN Services](#) on page 201, [Updating Dependent Services](#) on page 202, and [Configuring Alternatives Priority](#) on page 202) on this page.

The Activity Monitor role collects information about activities run by the MapReduce service. If MapReduce is not being used and the reporting data is no longer required, then the Activity Monitor role and database can be removed:

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Instances** tab.
3. Select checkbox for Activity Monitor, select **Actions for Selected > Stop**, and click **Stop** to confirm.
4. Select checkbox for Activity Monitor, select **Actions for Selected > Delete**, and click **Delete** to confirm.
5. Manage the Activity Monitor database. The example below is for a MySQL backend database:
 - a. Verify the Activity Monitor database:

```
mysql> show databases;
+-----+
| Database |
+-----+
| amon     |
+-----+
```

- a. Back up the database:

```
$ mysqldump -uroot -pcloudera amon > /safe_backup_directory/amon.sql
```

- a. Drop the database:

```
mysql> drop database amon;
```

Once you have migrated to YARN and deleted the MapReduce service, you can remove local data from each TaskTracker node. The `mapred.local.dir` parameter is a directory on the local filesystem of each TaskTracker that contains temporary data for MapReduce. Once the service is stopped, you can remove this directory to free disk space on each node.

For detailed information on migrating from MapReduce to YARN, see [Migrating from MapReduce 1 \(MRv1\) to MapReduce 2 \(MRv2, YARN\)](#).

Switching Between MapReduce and YARN Services

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

MapReduce and YARN use separate sets of configuration files. No files are removed or altered when you change to a different framework. To change from YARN to MapReduce (or vice versa):

1. (Optional) Configure the new MapReduce or YARN service.
2. [Update dependent services](#) to use the chosen framework.
3. Configure the [alternatives priority](#).
4. [Redeploy the Oozie ShareLib](#).
5. Redeploy the client configuration.
6. Start the framework service to switch to.
7. (Optional) Stop the unused framework service to free up the resources it uses.

Updating Dependent Services

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

When you change the MapReduce framework, the dependent services that must be updated to use the new framework are:

- Hive
- Sqoop 2
- Oozie

To update a service:

1. Go to the service.
2. Click the **Configuration** tab.
3. Select **Scope** > *service name*(Service Wide).
4. Select **Scope** > **All**.
5. Locate the **MapReduce Service** property and select the YARN or MapReduce service.
6. Click **Save Changes** to commit the changes.
7. Select **Actions** > **Restart**.

The Hue service is automatically reconfigured to use the same framework as Oozie and Hive. This cannot be changed.

To update the Hue service:

1. Go to the Hue service.
2. Select **Actions** > **Restart**.

Configuring Alternatives Priority

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

The alternatives priority property determines which service—MapReduce or YARN—is used by clients to run MapReduce jobs. The service with a higher value of the property is used. In CDH 4, the MapReduce service alternatives priority is set to 92 and the YARN service is set to 91. In CDH 5, the values are reversed; the MapReduce service alternatives priority is set to 91 and the YARN service is set to 92.

To configure the alternatives priority:

1. Go to the MapReduce or YARN service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Gateway Default Group**.
4. Select **Category** > **All**.
5. Type `Alternatives` in **Search** box.
6. In the **Alternatives Priority** property, set the priority value.
7. Click **Save Changes** to commit the changes.
8. Redeploy the client configuration.

Managing MapReduce

For an overview of computation frameworks, and their usage and restrictions, and common tasks, see [Managing MapReduce and YARN](#) on page 200.

Configuring the MapReduce Scheduler

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

The MapReduce service is configured by default to use the FairScheduler. You can change the scheduler type to FIFO or Capacity Scheduler. You can also modify the Fair Scheduler and Capacity Scheduler configuration. For further information on schedulers, see [Schedulers](#) on page 236.

Configuring the Task Scheduler Type

1. Go to the MapReduce service.
2. Click the **Configuration** tab.
3. Select **Scope > JobTracker**.
4. Select **Category > Classes**.
5. In the **Task Scheduler** property, select a scheduler.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Restart the JobTracker to apply the new configuration:
 - a. Click the **Instances** tab.
 - b. Click the **JobTracker** role.
 - c. Select **Actions for Selected > Restart**.

Modifying the Scheduler Configuration

1. Go to the MapReduce service.
2. Click the **Configuration** tab.
3. Select **Scope > JobTracker**.
4. Select **Category > Jobs**.
5. Modify the configuration properties.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Restart the JobTracker to apply the new configuration:
 - a. Click the **Instances** tab.
 - b. Click the **JobTracker** role.
 - c. Select **Actions for Selected > Restart**.

Configuring the MapReduce Service to Save Job History

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

Normally job history is saved on the host on which the JobTracker is running. You can configure JobTracker to write information about every job that completes to a specified HDFS location. By default, the information is retained for 7 days.

Enabling Map Reduce Job History To Be Saved to HDFS

1. Create a folder in HDFS to contain the history information. When creating the folder, set the owner and group to `mapred:hadoop` with permission setting 775.
2. Go to the MapReduce service.
3. Click the **Configuration** tab.
4. Select **Scope > JobTracker**.
5. Select **Category > Paths**.
6. Set the **Completed Job History Location** property to the location that you created in [step 1](#).

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Setting the Job History Retention Duration

1. Select the **JobTracker Default Group** category.
2. Set the **Job History Files Maximum Age** property (`mapreduce.jobhistory.max-age-ms`) to the length of time (in milliseconds, seconds, minutes, or hours) that you want job history files to be kept.
3. Restart the MapReduce service.

The Job History Files Cleaner runs at regular intervals to check for job history files that are ready to be deleted. By default, the interval is 24 hours. To change the frequency with which the Job History Files Cleaner runs:

1. Select the **JobTracker Default Group** category.
2. Set the **Job History Files Cleaner Interval** property (`mapreduce.jobhistory.cleaner.interval`) to the desired frequency (in milliseconds, seconds, minutes, or hours).
3. Restart the MapReduce service.

Configuring Client Overrides

A configuration property qualified with **(Client Override)** is a server-side setting that ignores any value a client tries to set for that property. It performs the same role as its unqualified counterpart, and applies the configuration to the service with the setting `<final>true</final>`.

For example, if you set the Map task heap property to 1 GB in the job configuration code, but the service's heap property qualified with (Client Override) is set to 500 MB, then 500 MB is applied.

Managing YARN

For an overview of computation frameworks, and their usage and restrictions, and common tasks, see [Managing MapReduce and YARN](#) on page 200.

Adding the YARN Service

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

1. On the **Home > Status** tab, click



to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.

2. Click the **YARN (MR2 Included)** radio button and click **Continue**.
3. Select the radio button next to the services on which the new service should depend. All services must depend on the *same* ZooKeeper service. Click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

Configuring Memory Settings for YARN and MRv2

The memory configuration for YARN and MRv2 memory is important to get the best performance from your cluster. Several different settings are involved. The table below shows the default settings, as well as the settings that Cloudera recommends, for each configuration option. See [Managing MapReduce and YARN](#) on page 200 for more configuration specifics; and, for detailed tuning advice with sample configurations, see [Tuning YARN](#) on page 277.

Table 3: YARN and MRv2 Memory Configuration

Cloudera Manager Property Name	CDH Property Name	Default Configuration	Cloudera Tuning Guidelines
Container Memory Minimum	yarn.scheduler.minimum-allocation-mb	1 GB	0
Container Memory Maximum	yarn.scheduler.maximum-allocation-mb	64 GB	amount of memory on largest node
Container Memory Increment	yarn.scheduler.increment-allocation-mb	512 MB	Use a fairly large value, such as 128 MB
Container Memory	yarn.nodemanager.resource.memory-mb	8 GB	8 GB
Map Task Memory	mapreduce.map.memory.mb	1 GB	1 GB
Reduce Task Memory	mapreduce.reduce.memory.mb	1 GB	1 GB
Map Task Java Opts Base	mapreduce.map.java.opts	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m
Reduce Task Java Opts Base	mapreduce.reduce.java.opts	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m
ApplicationMaster Memory	yarn.app.mapreduce.am.resource.mb	1 GB	1 GB
ApplicationMaster Java Opts Base	yarn.app.mapreduce.am.command-opts	-Djava.net.preferIPv4Stack=true	-Djava.net.preferIPv4Stack=true -Xmx768m

Configuring Directories

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

Creating the Job History Directory

When adding the YARN service, the **Add Service** wizard automatically creates a job history directory. If you quit the **Add Service** wizard or it does not finish, you can create the directory outside the wizard:

1. Go to the YARN service.
2. Select **Actions > Create Job History Dir**.
3. Click **Create Job History Dir** again to confirm.

Creating the NodeManager Remote Application Log Directory

When adding the YARN service, the **Add Service** wizard automatically creates a remote application log directory. If you quit the **Add Service** wizard or it does not finish, you can create the directory outside the wizard:

1. Go to the YARN service.
2. Select **Actions** > **Create NodeManager Remote Application Log Directory**.
3. Click **Create NodeManager Remote Application Log Directory** again to confirm.

Importing MapReduce Configurations to YARN


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



Warning: In addition to importing configuration settings, the import process:

- Configures services to use YARN as the MapReduce computation framework instead of MapReduce.
- Overwrites existing YARN configuration and role assignments.

When you upgrade from CDH 4 to CDH 5, you can import MapReduce configurations to YARN as part of the upgrade wizard. If you do not import configurations during upgrade, you can manually import the configurations at a later time:

1. Go to the YARN service page.
2. Stop the YARN service.
3. Select **Actions** > **Import MapReduce Configuration**. The import wizard presents a warning letting you know that it will import your configuration, restart the YARN service and its dependent services, and update the client configuration.
4. Click **Continue** to proceed. The next page indicates some additional configuration required by YARN.
5. Verify or modify the configurations and click **Continue**. The Switch Cluster to MR2 step proceeds.
6. When all steps have been completed, click **Finish**.
7. (Optional) Remove the MapReduce service.
 - a. Click the **Home** tab.
 - b. In the MapReduce row, right-click  and select **Delete**. Click **Delete** to confirm.
8. Recompile JARs used in MapReduce applications. For further information, see [For MapReduce Programmers: Writing and Running Jobs](#).

Configuring the YARN Scheduler

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

The YARN service is configured by default to use the FairScheduler. You can change the scheduler type to FIFO or Capacity Scheduler. You can also modify the Fair Scheduler and Capacity Scheduler configuration. For further information on schedulers, see [Schedulers](#) on page 236.

Configuring the Scheduler Type

1. Go to the YARN service.
2. Click the **Configuration** tab.
3. Select **Scope** > **ResourceManager**.
4. Select **Category** > **Main**.
5. Select a scheduler class.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** to commit the changes.
7. Restart the YARN service.

Modifying the Scheduler Configuration

1. Go to the YARN service.
2. Click the **Configuration** tab.
3. Click the **ResourceManager Default Group** category.
4. Select **Scope > ResourceManager**.
5. Type `Scheduler` in the Search box.
6. Locate a property and modify the configuration.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

Dynamic Resource Management

In addition to the [static resource management](#) available to all services, the YARN service also supports dynamic management of its static allocation. See [Dynamic Resource Pools](#) on page 242.

Configuring YARN for Long-running Applications

On a secure cluster, long-running applications such as Spark Streaming jobs will need additional configuration since the default settings only allow the `hdfe` user's delegation tokens a maximum lifetime of 7 days, which is not always sufficient. For instructions on how to work around this issue, see [Configuring YARN for Long-running Applications](#).

Task Process Exit Codes

All YARN tasks on the NodeManager are executed in a JVM. When a task executes successfully, the exit code is 0. Exit codes of 0 are not logged, as they are the expected result. Any non-zero exit code is logged as an error. The non-zero exit code is reported by the NodeManager as an error in the child process. The NodeManager itself is not affected by the error.

There are multiple reasons why the task JVM might exit with a non-zero code, though there is no exhaustive list. Exit codes can be split into two categories:

1. Set by the JVM based on the OS signal received by the JVM
2. Directly set in the code

Signal-related exit codes When the OS sends a signal to the JVM, the JVM handles the signal, which could cause the JVM to exit. Not all signals cause the JVM to exit. Exit codes for OS signals have a value between 128 and 160. Logs show non-zero status codes without further explanation.

Two exit values that typically do not require investigation are 137 and 143. These values are logged when the JVM is killed by the NodeManager or the OS. The NodeManager might kill a JVM due to task pre-emption (if that is configured) or speculative execution. The OS might kill the JVM when the JVM exceeds system limits like CPU time. You should investigate these codes if they appear frequently, as they might indicate a misconfiguration or a structural problem with regard to resources.

Exit code 154 is used in `RecoveredContainerLaunch#call` to indicate containers that were lost between NodeManager restarts without an exit code being recorded. This is usually a bug, and requires investigation.

Other exit codes The JVM might exit if there is an unrecoverable error while executing a task. The exit code and the message logged should provide more detail. A Java stack trace might also be logged as part of the exit. These exits should be investigated further to discover a root cause.

In the case of a streaming MapReduce job, the exit code of the JVM is the same as the mapper or reducer in use. The mapper or reducer can be a shell script or Python script. This means that the underlying script dictates the exit code: in streaming jobs, you should take this into account during your investigation.

Managing Oozie

This section describes tasks for managing Oozie.

Configuring Oozie for High Availability

In CDH 5, you can configure multiple active Oozie servers against the same database. Oozie high availability is “active-active” or “hot-hot” so that both Oozie servers are active at the same time, with no failover. High availability for Oozie is supported in both MRv1 and MRv2 (YARN).

Requirements

The requirements for Oozie high availability are:

- Multiple active Oozie servers, preferably identically configured.
- JDBC JAR in the same location across all Oozie hosts (for example, `/var/lib/oozie/`).
- External database that supports multiple concurrent connections, preferably with HA support. The default Derby database does not support multiple concurrent connections.
- ZooKeeper ensemble with distributed locks to control database access, and service discovery for log aggregation.
- Load balancer (preferably with HA support, for example [HAProxy](#)), virtual IP, or round-robin DNS to provide a single entry point (of the multiple active servers), and for callbacks from the Application Master or JobTracker.

For information on setting up SSL communication with Oozie HA enabled, see [Additional Considerations when Configuring SSL for Oozie HA](#).

Configuring Oozie High Availability Using Cloudera Manager

Minimum Required Role: [Full Administrator](#)



Important: Enabling or disabling high availability makes the previous monitoring history unavailable.

Enabling Oozie High Availability

1. Ensure that the [requirements](#) are satisfied.
2. In the Cloudera Manager Admin Console, go to the Oozie service.
3. Select **Actions** > **Enable High Availability** to see eligible Oozie server hosts. The host running the current Oozie server is not eligible.
4. Select the host on which to install an additional Oozie server and click **Continue**.
5. Specify the host and port of the Oozie load balancer, and click **Continue**. Cloudera Manager stops the Oozie servers, adds another Oozie server, initializes the Oozie server High Availability state in ZooKeeper, configures Hue to reference the Oozie load balancer, and restarts the Oozie servers and dependent services.

Disabling Oozie High Availability

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Select **Actions** > **Disable High Availability** to see all hosts currently running Oozie servers.
3. Select the one host to run the Oozie server and click **Continue**. Cloudera Manager stops the Oozie service, removes the additional Oozie servers, configures Hue to reference the Oozie service, and restarts the Oozie service and dependent services.

Configuring Oozie High Availability Using the Command Line

For installation and configuration instructions for configuring Oozie HA using the command line, see <https://archive.cloudera.com/cdh5/cdh/5/oozie>.

Adding the Oozie Service Using Cloudera Manager

The Oozie service can be automatically installed and started during your installation of CDH with Cloudera Manager.

You can also install Oozie manually with the **Add Service** wizard in Cloudera Manager. The wizard configures and starts Oozie and its dependent services. See [Adding a Service](#) on page 36 for instructions.



Note: If your instance of Cloudera Manager uses an external database, you must also configure Oozie with an external database. See [Configuring an External Database for Oozie](#).

Redeploying the Oozie ShareLib

Some Oozie actions – specifically DistCp, Streaming, Pig, Sqoop, and Hive – require external JAR files in order to run. Instead of having to keep these JAR files in each workflow's `lib` folder, or forcing you to manually manage them using the `oozie.libpath` property on every workflow using one of these actions, Oozie provides the ShareLib. The ShareLib behaves very similarly to `oozie.libpath`, except that it's specific to the aforementioned actions and their required JARs.

Redeploying the Oozie ShareLib Using Cloudera Manager

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

When you upgrade CDH or [switch between MapReduce and YARN](#) computation frameworks, redeploy the Oozie ShareLib as follows:

1. Go to the Oozie service.
2. Select **Actions > Stop**.
3. Select **Actions > Install Oozie ShareLib**.
4. Select **Actions > Start**.

Redeploying the Oozie ShareLib Using the Command Line

See [Installing the Oozie Shared Library in Hadoop HDFS](#).

Configuring Oozie Data Purge Settings Using Cloudera Manager

All Oozie workflows older than 30 days are purged from the database by default. However, actions associated with long-running coordinators do not purge until the coordinators complete. If, for example, you schedule a coordinator to run for a year, all those actions remain in the database for the year.

You can change your Oozie configuration to control when data is purged to improve performance, reduce database disk usage, or keep the history for a longer period of time. Limiting the size of the Oozie database can also improve performance during upgrades.

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the **Configuration** tab.
3. Type `purge` in the Search box.
4. Set the following properties as required for your environment:
 - **Enable Purge for Long-Running Coordinator Jobs**
Select this property to enable purging of long-running coordinator jobs for which the workflow jobs are older than the value you set for the **Days to Keep Completed Workflow Jobs** property.
 - **Days to Keep Completed Workflow Jobs**
 - **Days to Keep Completed Coordinator Jobs**
 - **Days to Keep Completed Bundle Jobs**
5. Click **Save Changes** to commit the changes.
6. Select **Actions > Restart** to restart the Oozie Service.

Adding Schema to Oozie Using Cloudera Manager

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

This page explains how to manually add a schema (official or custom) with Cloudera Manager.

Cloudera Manager 5 automatically configures Oozie with all available official schemas, and corresponding tables, per the latest CDH 5.x release. For all versions of CDH 4.x, Cloudera Manager configures Oozie with the CDH 4.0.0 schema, even if you are using a higher version of CDH 4.x.

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the **Configuration** tab.
3. Select **Scope > Oozie Server**.
4. Select **Category > Advanced**.
5. Locate the **Oozie SchemaService Workflow Extension Schemas** property or search for it by typing its name in the Search box.
6. Enter the desired schema from [Table 4: Oozie Schema - CDH 5](#) on page 210 or [Table 5: Oozie Schema - CDH 4](#) on page 211, appending `.xsd` to each entry.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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



Note: Releases are only included in the following tables if a schema was added or removed. If a release is not in the table, it has the same set of schemas as the previous release that is in the table.

Table 4: Oozie Schema - CDH 5

	CDH 5.4.0	CDH 5.2.0	CDH 5.1.0	CDH 5.0.1	CDH 5.0.0
distcp	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1 distcp-action-0.2
email	email-action-0.1 email-action-0.2	email-action-0.1 email-action-0.2	email-action-0.1 email-action-0.2	email-action-0.1	email-action-0.1
hive	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5
HiveServer2	hive2-action-0.1	hive2-action-0.1			
oozie-bundle	hive2-action-0.1	hive2-action-0.1	oozie-bundle-0.1 oozie-bundle-0.2	oozie-bundle-0.1 oozie-bundle-0.2	oozie-bundle-0.1 oozie-bundle-0.2
oozie-coordinator	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4
oozie-sla	oozie-sla-0.1 oozie-sla-0.2	oozie-sla-0.1 oozie-sla-0.2	oozie-sla-0.1 oozie-sla-0.2	oozie-sla-0.1 oozie-sla-0.2	oozie-sla-0.1 oozie-sla-0.2
oozie-workflow	oozie-workflow-0.1 oozie-workflow-0.2	oozie-workflow-0.1 oozie-workflow-0.2	oozie-workflow-0.1 oozie-workflow-0.2	oozie-workflow-0.1 oozie-workflow-0.2	oozie-workflow-0.1 oozie-workflow-0.2

	CDH 5.4.0	CDH 5.2.0	CDH 5.1.0	CDH 5.0.1	CDH 5.0.0
	oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.5
shell	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3
spark	spark-action-0.1				
sqoop	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4
ssh	ssh-action-0.1 ssh-action-0.2	ssh-action-0.1 ssh-action-0.2	ssh-action-0.1 ssh-action-0.2	ssh-action-0.1 ssh-action-0.2	ssh-action-0.1 ssh-action-0.2

Table 5: Oozie Schema - CDH 4

	CDH 4.3.0	CDH 4.2.0	CDH 4.1.0	CDH 4.0.0
distcp	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1 distcp-action-0.2	distcp-action-0.1	distcp-action-0.1
email	email-action-0.1	email-action-0.1	email-action-0.1	email-action-0.1
hive	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5	hive-action-0.2 hive-action-0.3 hive-action-0.4	hive-action-0.2 hive-action-0.3 hive-action-0.4	hive-action-0.2
oozie-bundle	oozie-bundle-0.1 oozie-bundle-0.2	oozie-bundle-0.1 oozie-bundle-0.2	oozie-bundle-0.1 oozie-bundle-0.2	oozie-bundle-0.1
oozie-coordinator	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3 oozie-coordinator-0.4	oozie-coordinator-0.1 oozie-coordinator-0.2 oozie-coordinator-0.3
oozie-sla	oozie-sla-0.1	oozie-sla-0.1	oozie-sla-0.1	oozie-sla-0.1
oozie-workflow	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5

	CDH 4.3.0	CDH 4.2.0	CDH 4.1.0	CDH 4.0.0
	oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5	oozie-workflow-0.3 oozie-workflow-0.4	oozie-workflow-0.3 oozie-workflow-0.4	oozie-workflow-0.3
shell	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1 shell-action-0.2 shell-action-0.3	shell-action-0.1
sqoop	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2 sqoop-action-0.3 sqoop-action-0.4	sqoop-action-0.2
ssh	ssh-action-0.1 ssh-action-0.2	ssh-action-0.1	ssh-action-0.1	ssh-action-0.1

Enabling the Oozie Web Console

Enabling the Oozie Web Console Using Cloudera Manager

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

1. Download [ext-2.2](#).
2. Extract the contents of the file to `/var/lib/oozie/` on the same host as the Oozie Server.

For example:

```
unzip ext-2.2.zip -d /var/lib/oozie
chown -R oozie:oozie /var/lib/oozie/ext-2.2
```

After that the content of the directories is the following:

```
$ ls -ltr /var/lib/oozie/
total 984
drwxr-xr-x 9 oozie oozie 4096 Aug 4 2008 ext-2.2
drwxr-xr-x 2 oozie oozie 6 Nov 30 02:25 work
-rw-r--r-- 1 systest root 999635 Dec 2 07:32 mysql-connector-java.jar
drwxr-xr-x 5 oozie oozie 42 Dec 2 07:43 tomcat-deployment
$ ls -ltr /var/lib/oozie/ext-2.2/
total 1752
-rw-r--r-- 1 oozie oozie 893 Feb 24 2008 INCLUDE_ORDER.txt
drwxr-xr-x 33 oozie oozie 4096 Aug 4 2008 examples
drwxr-xr-x 4 oozie oozie 49 Aug 4 2008 resources
drwxr-xr-x 10 oozie oozie 148 Aug 4 2008 source
drwxr-xr-x 10 oozie oozie 120 Aug 4 2008 build
-rw-r--r-- 1 oozie oozie 87524 Aug 4 2008 ext-core.js
-rw-r--r-- 1 oozie oozie 163794 Aug 4 2008 ext-core-debug.js
-rw-r--r-- 1 oozie oozie 974145 Aug 4 2008 ext-all-debug.js
drwxr-xr-x 6 oozie oozie 55 Aug 4 2008 adapter
-rw-r--r-- 1 oozie oozie 11548 Aug 4 2008 CHANGES.html
-rw-r--r-- 1 oozie oozie 538956 Aug 4 2008 ext-all.js
-rw-r--r-- 1 oozie oozie 1513 Aug 4 2008 license.txt
drwxr-xr-x 4 oozie oozie 108 Aug 4 2008 docs
drwxr-xr-x 5 oozie oozie 94 Dec 3 02:27 air
```

3. In Cloudera Manager Admin Console, go to the Oozie service.
4. Locate the **Enable Oozie server web console** property or search for it by typing its name in the Search box.
5. Select **Enable Oozie server web console**.
6. To commit the changes, click **Save Changes**.

7. Restart the Oozie service.

Enabling the Oozie Web Console Using the Command Line

See [Enabling the Oozie Web Console](#).

Setting the Oozie Database Timezone

We recommend that you set the timezone in the Oozie database to GMT. Databases do not handle Daylight Saving Time (DST) shifts correctly. There might be problems if you run any Coordinators with actions scheduled to materialize during the one-hour period that gets lost in DST.

- To set the timezone in Derby, add the following to `CATALINA_OPTS` in the `oozie-env.sh` file:

```
-Duser.timezone=GMT
```

- To set the timezone just for Oozie in MySQL, add the following argument to `oozie.service.JPAAService.jdbc.url`:

```
useLegacyDatetimeCode=false&serverTimezone=GMT
```



Important: Changing the timezone on an existing Oozie database while Coordinators are already running might cause Coordinators to shift by the offset of their timezone from GMT one time after you make this change.

For more information about how to set your database's timezone, see your database's documentation.

Scheduling in Oozie Using Cron-like Syntax

Most Linux distributions include the [cron](#) utility, which is used for scheduling time-based jobs. For example, you might want cron to run a script that deletes your internet history once a week. This topic details how to schedule Oozie using Cron-like syntax.

Location

Set the scheduling information in the `frequency` attribute of the `coordinator.xml` file. A simple file looks like the following example. The `frequency` attribute and scheduling information appear in bold.

```
<coordinator-app name="MY_APP" frequency="30 14 * *
  ** start="2009-01-01T05:00Z" end="2009-01-01T06:00Z" timezone="UTC"
  xmlns="uri:oozie:coordinator:0.5">
  <action>
    <workflow>
      <app-path>hdfs://localhost:8020/tmp/workflows</app-path>
    </workflow>
  </action>
</coordinator-app>
```



Important: Before CDH 5 Oozie used fixed-frequency scheduling. You could only schedule according to a set amount of minutes or a set time configured in an EL (Expression Language) function. The cron-like syntax allows more flexibility.

Syntax and Structure

The cron-like syntax used by Oozie is a string with five space-separated fields:

- minute
- hour
- day-of-month

- month
- day-of-week

The structure takes the form of * * * * *. For example, 30 14 * * * means that the job runs at at 2:30 p.m. everyday. The minute field is set to 30, the hour field is set to 14, and the remaining fields are set to *.

Allowed Values and Special Characters

The following table describes special characters allowed and indicates in which fields they can be used.


Table 6: Special Characters

Character	Fields Allowed	Description
* (asterisk)	All	Match all values.
, (comma)	All	Specify multiple values.
- (dash)	All	Specify a range.
/ (forward slash)	All	Specify an increment.
? (question mark)	Day-of-month, day-of-week	Indicate no specific value (for example, if you want to specify one but not the other).
L	Day-of-month, day-of-week	Indicate the last day of the month or the last day of the week (Saturday). In the day-of-week field, 6L indicates the last Friday of the month.
W	Day-of-month	Indicate the nearest weekday to the given day.
# (pound sign)	Day-of-week	Indicate the <i>n</i> th day of the month

The following table summarizes the valid values for each field.

Field	Allowed Values	Allowed Special Characters
Minute	0-59	, - * /
Hour	0-23	, - * /
Day-of-month	0-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day-of-week	1-7 or SUN-SAT	, - * ? / L #

For more information about Oozie cron-like syntax, see [Cron syntax in coordinator frequency](#).



Important: Some cron implementations accept 0-6 as the range for days of the week. Oozie accepts 1-7 instead.

Scheduling Examples

The following examples show cron scheduling in Oozie. Oozie’s processing time zone is UTC. If you are in a different time zone, add to or subtract from the appropriate offset in these examples.

Run at the 30th minute of every hour

Set the minute field to 30 and the remaining fields to * so they match every value.

```
frequency="30 * * * *"
```

Run at 2:30 p.m. every day

Set the minute field to 30, the hour field to 14, and the remaining fields to *.

```
frequency="30 14 * * *"
```

Run at 2:30 p.m. every day in February

Set the minute field to 30, the hour field to 14, the day-of-month field to *, the month field to 2 (February), and the day-of-week field to *.

```
frequency="30 14 * 2 *"
```

Run every 20 minutes between 5:00-10:00 a.m. and between 12:00-2:00 p.m. on the fifth day of each month

Set the minute field to 0/20, the hour field to 5-9, 12-14, the day-of-month field to 0/5, and the remaining fields to *.

```
frequency="0/20 5-9,12-14 0/5 * *"
```

Run every Monday at 5:00 a.m.

Set the minute field to 0, the hour field to 5, the day-of-month field to ?, the month field to *, and the day-of-week field to MON.

```
frequency="0 5 ? * MON"
```



Note: If the ? was set to *, this expression would run the job every day at 5:00 a.m., not just Mondays.

Run on the last day of every month at 5:00 a.m.

Set the minute field to 0, the hour field to 5, the day-of-month field to L, the month field to *, and the day-of-week field to ?.

```
frequency="0 5 L * ?"
```

Run at 5:00 a.m. on the weekday closest to the 15th day of each month

Set the minute field to 0, the hour field to 5, the day-of-month field to 15W, the month field to *, and the day-of-week field to ?.

```
frequency="0 5 15W * ?"
```

Run every 33 minutes from 9:00-3:00 p.m. on the first Monday of every month

Set the minute field to 0/33, the hour field to 9-14, the day-of-week field to 2#1 (the first Monday), and the remaining fields to *.

```
frequency="0/33 9-14 ? * 2#1"
```

Run every hour from 9:00 a.m.-5:00 p.m. on weekdays

Set the minute field to 0, the hour field to 9-17, the day-of-month field to ?, the month field to *, and the day-of-week field to 2-6.

```
frequency="0 9-17 ? * 2-6"
```

Run on the second-to-last day of every month

Set the minute field to 0, the hour field to 0, the day-of-month field to L-1, the month field to *, and the day-of-week field to ?.

```
frequency="0 0 L-1 * ?"
```



Note: "L-1" means the second-to-last day of the month.

Oozie uses [Quartz](#), a job scheduler library, to parse the cron syntax. For more examples, go to the [CronTrigger Tutorial](#) on the Quartz website. Quartz has two fields (second and year) that Oozie does not support.

Managing Solr

You can install the Solr service through the Cloudera Manager installation wizard, using either parcels or packages. See [Installing Search](#).

You can elect to have the service created and started as part of the Installation wizard. If you elect not to create the service using the Installation wizard, you can use the **Add Service** wizard to perform the installation. The wizard will automatically configure and start the dependent services and the Solr service. See [Adding a Service](#) on page 36 for instructions.

For further information on the Solr service, see [Cloudera Search Guide](#).

The following sections describe how to configure other CDH components to work with the Solr service.

Configuring the Flume Morphline Solr Sink for Use with the Solr Service

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

To use a Flume Morphline Solr sink, the Flume service must be running on your cluster. See the [Flume Near Real-Time Indexing Reference \(CDH 5\)](#) for information about the Flume Morphline Solr Sink and [Managing Flume](#) on page 76.

1. Go to the Flume service.
2. Click the **Configuration** tab.
3. Select **Scope > Agent**
4. Select **Category > Flume-NG Solr Sink**.
5. Edit the following settings, which are templates that you must modify for your deployment:
 - **Morphlines File** (`morphlines.conf`) - Configures Morphlines for Flume agents. You must use `$ZK_HOST` in this field instead of specifying a ZooKeeper quorum. Cloudera Manager automatically replaces the `$ZK_HOST` variable with the correct value during the Flume configuration deployment.
 - **Custom MIME-types File** (`custom-mimetypes.xml`) - Configuration for the `detectMimeTypes` command. See the [Cloudera Morphlines Reference Guide](#) for details on this command.
 - **Grok Dictionary File** (`grok-dictionary.conf`) - Configuration for the `grok` command. See the [Cloudera Morphlines Reference Guide](#) for details on this command.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

Once configuration is complete, Cloudera Manager automatically deploys the required files to the Flume agent process directory when it starts the Flume agent. Therefore, you can reference the files in the [Flume agent configuration](#) using their relative path names. For example, you can use the name `morphlines.conf` to refer to the location of the Morphlines configuration file.

Deploying Solr with Hue

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

In CDH 4.3 and lower, in order to use Solr with Hue, you must update the URL for the Solr Server in the Hue Server advanced configuration snippet.

1. Go to the **Hue** service.
2. Click the **Configuration** tab.
3. Type the word "snippet" in the Search box.

A set of Hue advanced configuration snippet properties displays.

4. Add information about your Solr host to the **Hue Server Configuration Advanced Configuration Snippet for hue_safety_valve_server.ini** property. For example, if your hostname is `SOLR_HOST`, you might add the following:

```
[search]
## URL of the Solr Server
solr_url=http://SOLR_HOST:8983/solr
```

5. Click **Save Changes** to save your advanced configuration snippet changes.
6. Restart the Hue Service.



Important: If you are using parcels with CDH 4.3, you must register the "hue-search" application manually or access will fail. You do not need to do this if you are using CDH 4.4 and higher.

1. Stop the Hue service.
2. From the command line do the following:

```
cd /opt/cloudera/parcels/CDH 4.3.0-1.cd4.3.0.pXXX/share/hue
```

(Substitute your own local repository path for the `/opt/cloudera/parcels/...` if yours is different, and specify the appropriate name of the CDH 4.3 parcel that exists in your repository.)

```
./build/env/bin/python ./tools/app_reg/app_reg.py
--install
/opt/cloudera/parcels/SOLR-0.9.0-1.cd4.3.0.pXXX/share/hue/apps/search
```

```
sed -i 's/\.\./apps/..\./..\./..\./..\./apps/g'
./build/env/lib/python2.X/site-packages/hue.pth
```

where `python2.X` should be the version you are using (for example, `python2.4`).

3. Start the Hue service.

Managing Spark

[Apache Spark](#) is a general framework for distributed computing that offers very high performance for both iterative and interactive processing. Spark exposes APIs for Java, Python, and Scala. Spark consists of Spark core and several related projects:

- [Spark SQL](#) - module for working with structured data. Allows you to seamlessly mix SQL queries with Spark programs.
- [Spark Streaming](#) - API that allows you to build scalable fault-tolerant streaming applications.
- [MLlib](#) - library that implements common [machine learning](#) algorithms.
- [GraphX](#) - API for graphs and graph-parallel computation.

Cloudera supports Spark core and Spark Streaming. Cloudera does not currently offer commercial support for Spark SQL (including DataFrames), MLlib, and GraphX.

To run applications distributed across a cluster, Spark requires a cluster manager. Cloudera supports two cluster managers: Spark Standalone and YARN. Cloudera does not support running Spark applications on Mesos. On Spark

Standalone, Spark application processes run on Spark Master and Worker roles. On YARN, Spark application processes run on YARN ResourceManager and NodeManager roles.

In CDH 5, Cloudera recommends running Spark applications on a [YARN](#) cluster manager instead of on a Spark Standalone cluster manager, for the following benefits:

- You can dynamically share and centrally configure the same pool of cluster resources among all frameworks that run on YARN.
- You can use [all the features of YARN schedulers](#) for categorizing, isolating, and prioritizing workloads.
- You choose the number of executors to use; in contrast, Spark Standalone requires each application to run an executor on every host in the cluster.
- Spark can run against Kerberos enabled Hadoop clusters and use secure authentication between its processes.

This section describes how to manage Spark services and run Spark applications.

Related Information

- [Monitoring Spark Applications](#)
- [Tuning Spark Applications](#) on page 277
- [Cloudera Spark forum](#)
- [Apache Spark documentation](#)

Managing Spark Using Cloudera Manager

Spark is available as two services: Spark and Spark (Standalone).

In Cloudera Manager 5.1 and lower, the Spark service runs a Spark Standalone cluster, which has Master and Worker roles.

In Cloudera Manager 5.2 and higher, the service that runs a Spark Standalone cluster has been renamed Spark (Standalone), and the Spark service runs Spark as a YARN application with only gateway roles. Both services have a [Spark History Server](#) role.

You can install, add, and start Spark through the Cloudera Manager Installation wizard using parcels. For more information, see [Installing Spark](#).

If you do not add the Spark service using the Installation wizard, you can use the **Add Service** wizard to create the service. The wizard automatically configures dependent services and the Spark service. For instructions, see [Adding a Service](#) on page 36.

When you upgrade from Cloudera Manager 5.1 or lower to Cloudera 5.2 or higher, Cloudera Manager *does not* migrate an existing Spark service, which runs Spark Standalone, to a Spark on YARN service.

For information on Spark applications, see [Spark Applications](#) on page 220.

How Spark Configurations are Propagated to Spark Clients

Because the Spark 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 Spark 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.

Managing Spark Standalone Using the Command Line

This section describes how to configure and start Spark Standalone services.

For information on installing Spark using the command line, see [Spark Installation](#). For information on configuring and starting the Spark History Server, see [Configuring and Running the Spark History Server Using the Command Line](#) on page 220.

For information on Spark applications, see [Spark Applications](#) on page 220.

Configuring Spark Standalone

Before running Spark Standalone, do the following on every host in the cluster:

- Edit `/etc/spark/conf/spark-env.sh` and change `hostname` in the last line to the name of the host where the Spark Master will run:

```
###
### === IMPORTANT ===
### Change the following to specify the Master host
###
export STANDALONE_SPARK_MASTER_HOST=`hostname`
```

- Optionally, edit other configuration options:
 - `SPARK_MASTER_PORT` / `SPARK_MASTER_WEBUI_PORT` and `SPARK_WORKER_PORT` / `SPARK_WORKER_WEBUI_PORT`, to use non-default ports
 - `SPARK_WORKER_CORES`, to set the number of cores to use on this machine
 - `SPARK_WORKER_MEMORY`, to set how much memory to use (for example: 1000 MB, 2 GB)
 - `SPARK_WORKER_INSTANCE`, to set the number of worker processes per node
 - `SPARK_WORKER_DIR`, to set the working directory of worker processes

Starting and Stopping Spark Standalone Clusters

To start Spark Standalone clusters:

1. On one host in the cluster, start the Spark Master:

```
$ sudo service spark-master start
```

You can access the Spark Master UI at `spark_master:18080`.

2. On all the other hosts, start the workers:

```
$ sudo service spark-worker start
```

To stop Spark, use the following commands on the appropriate hosts:

```
$ sudo service spark-worker stop
$ sudo service spark-master stop
```

Service logs are stored in `/var/log/spark`.

Managing the Spark History Server

The Spark History Server displays information about the history of completed Spark applications. For further information, see [Monitoring Spark Applications](#).

For instructions for configuring the Spark History Server to use Kerberos, see [Spark Authentication](#).

Adding the Spark History Server Using Cloudera Manager

By default, the Spark (Standalone) service does not include a History Server. To configure applications to store history, on Spark clients, set `spark.eventLog.enabled` to `true` before starting the application.

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

To add the History Server:

1. Go to the Spark service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Select a host in the column under **History Server**, and then click **OK**.

5. Click **Continue**.
6. Check the checkbox next to the History Server role.
7. Select **Actions for Selected** > **Start** and click **Start**.
8. Click **Close** when the action completes.

Configuring and Running the Spark History Server Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Create the `/user/spark/applicationHistory/` directory in HDFS and set ownership and permissions as follows:

```
$ sudo -u hdfs hadoop fs -mkdir /user/spark
$ sudo -u hdfs hadoop fs -mkdir /user/spark/applicationHistory
$ sudo -u hdfs hadoop fs -chown -R spark:spark /user/spark
$ sudo -u hdfs hadoop fs -chmod 1777 /user/spark/applicationHistory
```

2. On hosts from which you will launch Spark jobs, do the following:

- a. Create `/etc/spark/conf/spark-defaults.conf`:

```
cp /etc/spark/conf/spark-defaults.conf.template /etc/spark/conf/spark-defaults.conf
```

- b. Add the following to `/etc/spark/conf/spark-defaults.conf`:

```
spark.eventLog.dir=/user/spark/applicationHistory
spark.eventLog.enabled=true
```

This causes Spark applications to write their history to the directory that the History Server reads.

- c. On one host, start the History Server:

```
$ sudo service spark-history-server start
```

To link the YARN ResourceManager directly to the Spark History Server, set the `spark.yarn.historyServer.address` property in `/etc/spark/conf/spark-defaults.conf`:

```
spark.yarn.historyServer.address=http://spark_history_server:history_port
```

By default, `history_port` is 18088.

Spark Applications

Spark applications are similar to MapReduce jobs. Each application is a self-contained computation that runs user-supplied code to compute a result. As with MapReduce jobs, Spark applications can make use of the resources of multiple hosts.

In MapReduce, the highest-level unit of computation is a **job**. The system loads the data, applies a map function, shuffles it, applies a reduce function, and writes it back out to persistent storage. In Spark, the highest-level of computation is an **application**, which can run multiple jobs, in sequence or in parallel. A Spark job can consist of more stages than just a single map and reduce.

To manage the job flow and schedule tasks, Spark relies on a **driver** process. Typically, this driver process is the same as the client process used to initiate the job, although in YARN mode, the driver can run on the cluster. In contrast, in MapReduce, the client process can terminate and the job will continue running.

A Spark application corresponds to an instance of the `SparkContext` class. An application can be used for a single batch job, an interactive session with multiple jobs spaced apart, or a long-lived server continually satisfying requests. Unlike MapReduce, an application will have processes, called **executors**, running on its behalf even when it's not running any jobs. This approach enables data storage in memory for quick access, as well as lightning-fast task startup time.

Spark Terminology

resilient distributed dataset (RDD)

The core programming abstraction in Spark, consisting of a fault-tolerant collection of elements that can be operated on in parallel.

partition

A subset of the elements in an RDD. Partitions define the unit of parallelism; Spark processes elements within a partition in sequence and multiple partitions in parallel. When Spark reads a file from HDFS, it creates a single partition for a single input split. It returns a single partition for a single block of HDFS (but the split between partitions is on line split, not the block split), unless you have a compressed text file. In case of compressed file you would get a single partition for a single file (as compressed text files are not splittable).

application

A job, sequence of jobs, or a long-running service issuing new commands as needed or an interactive exploration session.

application JAR

A JAR containing a Spark application. In some cases you can use an "Uber" JAR containing your application along with its dependencies. The JAR should never include Hadoop or Spark libraries, however, these will be added at runtime.

cluster manager

An external service for acquiring resources on the cluster: Spark Standalone or YARN.

job

A parallel computation consisting of multiple tasks that gets spawned in response to a Spark action.

task

A unit of work on a partition of a distributed dataset. Also referred to as a **stage**.

driver

Process that represents the application session. The driver is responsible for converting the application to a directed graph of individual steps to execute on the cluster. There is one driver per application.

executor

A process that serves a Spark application. An executor runs multiple tasks over its lifetime, and multiple tasks concurrently. A host may have several Spark executors and there are many hosts running Spark executors for each application.

deploy mode

Identifies where the driver process runs. In **client mode**, the submitter launches the driver outside of the cluster. In **cluster mode**, the framework launches the driver inside the cluster. Client mode is simpler, but cluster mode allows you to log out after starting a Spark application without terminating the application.

Spark Standalone

A model of running Spark applications in which a Master daemon coordinates the efforts of Worker daemons, which run the executors.

Spark on YARN

A model of running Spark applications in which the YARN ResourceManager performs the functions of the Spark Master. The functions of the Workers are performed by the YARN NodeManagers, which run the executors.

ApplicationMaster

A YARN role responsible for negotiating resource requests made by the driver and finding a set of containers in which to run the Spark application. There is one ApplicationMaster per application.

Configuring Spark Applications

You can specify application configuration parameters as follows:

- Pass them directly to the [SparkConf](#) that is used to create the [SparkContext](#) in your Spark application; for example:

```
val conf = new SparkConf().set("spark.dynamicAllocation.initialExecutors", "5")
val sc = new SparkContext(conf)
```

- To avoid hard-coding them in the application:
 - Pass them using the `--conf` command-line switch; for example:

```
spark-submit \
  --class com.cloudera.example.YarnExample \
  --master yarn \
  --deploy-mode cluster \
  --conf "spark.eventLog.dir=hdfs:///user/spark/eventlog" \
  lib/yarn-example.jar \
  10
```

- Specify them in `spark-defaults.conf`. This file contains lines in the form: "property value". You can create a comment by putting a hash mark (#) at the beginning of a line. You cannot add comments to the end or middle of a line.

Here is an example of a `spark-defaults.conf` file:

```
spark.master      spark://mysparkmaster.acme.com:7077
spark.eventLog.enabled      true
spark.eventLog.dir      hdfs:///user/spark/eventlog
# Set spark executor memory
spark.executor.memory    2g
spark.logConf            true
```

It is a good idea to put configuration properties that you want to use for every application into `spark-defaults.conf`. See [Application Properties](#) for more information.

The order of precedence in configuration parameters is:

1. Parameters passed to `SparkConf`.
2. Arguments passed to `spark-submit`, `spark-shell`, or `pyspark`.
3. Properties set in `spark-defaults.conf`.

For more information, see [Running Spark on YARN](#) and [Spark Configuration](#).

Specifying Properties in spark-defaults.conf Using Cloudera Manager

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

1. Go to the Spark service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Spark Service_Name Service-Wide**.
4. Select **Category** > **Advanced**.
5. Locate the **Spark Client Advanced Configuration Snippet (Safety Valve) for spark-conf/spark-defaults.conf** property.
6. Specify properties described in [Application Properties](#).

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

7. Click **Save Changes** to commit the changes.
8. Deploy the client configuration.

Specifying Properties in `spark-defaults.conf` Using the Command Line

To specify properties in `spark-defaults.conf` using the command line, edit the file `$SPARK_HOME/conf/spark-defaults.conf`.

Building Spark Applications

Best practices when building Spark applications include:

- Compiling against the same version of Spark that you are running against, to avoid compatibility issues. For example, do not assume that applications compiled against Spark 0.9 will run on Spark 1.0 without recompiling. In addition, some applications compiled under Spark 0.9 or earlier will need changes to their source code to compile under Spark 1.0. Applications that compile under Spark 1.0 should compile under all future versions. "Uber" JARs must always be built against the version of Spark you intend to run (see [Apache Spark Known Issues](#)).
- Building a single assembly JAR that includes all the dependencies. Exclude Spark and Hadoop classes from the assembly JAR, because they are already on the cluster, and part of the runtime classpath. In Maven, you can mark the Spark and Hadoop dependencies as `provided`.

Enabling Fault-Tolerant Processing in Spark Streaming

If the driver host for a Spark Streaming application fails, it can lose data that has been received, but not yet processed. To ensure that no data is lost, Spark can write out incoming data to HDFS as it is received and use this data to recover state in the event of a failure. This feature, called **Spark Streaming recovery**, was introduced in CDH 5.3 as a Beta feature. Spark Streaming recovery is supported for production use in CDH 5.4.

To enable Spark Streaming recovery:

1. Set the `spark.streaming.receiver.writeAheadLog.enable` parameter to `true` in the `SparkConf` object:

```
sparkConf.set("spark.streaming.receiver.writeAheadLog.enable", "true")
```

2. Create a `StreamingContext` instance using this `SparkConf`, and specify a checkpoint directory.
3. Use the `getOrCreate` method in `StreamingContext` to either create a new context or recover from an old context from the checkpoint directory:

```
// Function to create and setup a new StreamingContext
def functionToCreateContext(): StreamingContext = {
  val conf = new SparkConf()
  sparkConf.set("spark.streaming.receiver.writeAheadLog.enable", "true ")
  val ssc = new StreamingContext(sparkConf,...) // new context

  val kafkaStream = KafkaUtils.createStream(...)
  // Do some transformations on the stream...and write it out etc.

  ssc.checkpoint(checkpointDirectory) // set checkpoint directory
  ssc
}

// Get StreamingContext from checkpoint data or create a new one
val context = StreamingContext.getOrCreate(checkpointDirectory, functionToCreateContext
_)
```

For further information, see [Checkpointing](#).

To prevent data loss if a receiver fails, receivers must be able to replay data from the original data sources if required.

- The Kafka receiver automatically replays if the `spark.streaming.receiver.writeAheadLog.enable` parameter is set to `true`.
- The receiver-less Direct Kafka `DStream` does not require the `spark.streaming.receiver.writeAheadLog.enable` parameter, and can function without data loss even without Streaming recovery.
- Both the Flume receivers that come packaged with Spark replay the data automatically on receiver failure.

See [Spark Streaming + Kafka Integration Guide](#) and [Spark Streaming + Flume Integration Guide](#).

Using Spark with HBase

A common use case is to use Spark to process data that is destined for HBase. See [Importing Data Into HBase](#) on page 105.

Running Spark Applications

You can run Spark applications locally or distributed across a cluster, either by using an interactive shell or by submitting an application.

To run applications distributed across a cluster, Spark requires a cluster manager. Cloudera supports two cluster managers: Spark Standalone and YARN. Cloudera does not support running Spark applications on Mesos. On Spark Standalone, Spark application processes run on Spark Master and Worker roles. On YARN, Spark application processes run on YARN ResourceManager and NodeManager roles.

In CDH 5, Cloudera recommends running Spark applications on a [YARN](#) cluster manager instead of on a Spark Standalone cluster manager, for the following benefits:

- You can dynamically share and centrally configure the same pool of cluster resources among all frameworks that run on YARN.
- You can use [all the features of YARN schedulers](#) for categorizing, isolating, and prioritizing workloads.
- You choose the number of executors to use; in contrast, Spark Standalone requires each application to run an executor on every host in the cluster.
- Spark can run against Kerberos enabled Hadoop clusters and use secure authentication between its processes.

For information on monitoring Spark applications, see [Monitoring Spark Applications](#).

Running Spark Applications Interactively

You can run a Spark application interactively using the the Scala `spark-shell` or Python `pyspark` shell application. For a complete list of [options](#), run `spark-shell` or Python `pyspark` with the `-h` flag.

For example, when the shell is running, you can perform a word-count application using the following code examples:

- Scala

```
val file = sc.textFile("hdfs://namenode:8020/path/to/input")
val counts = file.flatMap(line => line.split(" ")).map(word => (word, 1)).reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://namenode:8020/user/hdfs/output")
```

- Python

```
file = sc.textFile("hdfs://namenode:8020/path/to/input")
counts = file.flatMap(lambda line: line.split(" ")).map(lambda word: (word, 1)).reduceByKey(lambda v1,v2: v1 + v2)
counts.saveAsTextFile("hdfs://namenode:8020/user/hdfs/output")
```



Note: Some applications run in an interactive shell that have nested definitions may encounter a `Task not serializable` exception, because of a limitation in the way Scala compiles code. Cloudera recommends submitting these applications to a cluster.

Submitting Spark Applications

Using the `spark-submit` Script

You submit compiled Spark applications with the `spark-submit` script.

```
SPARK_HOME/bin/spark-submit \  
  --option value \  
  application jar | python file \  
  [application arguments]
```


[Example: Running SparkPi on YARN](#) on page 230 and [Example: Running SparkPi on Spark Standalone](#) on page 231 demonstrate how to run a sample application, `SparkPi`, which is packaged with Spark. It computes an approximation to the value of Pi.

Table 7: spark-submit Options

Option	Description
<code>application jar</code>	Path to a bundled JAR file including your application and all dependencies. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the URL must be globally visible inside your cluster; for example, an <code>hdfs://</code> path or a <code>file://</code> path that exists on all nodes.
<code>python file</code>	Path to a Python file containing your application. For the client deployment mode, the path must point to a local file. For the cluster deployment mode, the URL must be globally visible inside your cluster; for example, an <code>hdfs://</code> path or a <code>file://</code> path that exists on all nodes.
<code>--py-files python files</code>	Comma-separated list of <code>.zip</code> , <code>.egg</code> , or <code>.py</code> files to place on the <code>PYTHONPATH</code> for Python apps.
<code>--files files</code>	Comma-separated list of files to be placed in the working directory of each executor.
<code>application arguments</code>	Arguments passed to the main method of your main class, if any.

Command-Line Options

You specify command-line options using the form `--option value` instead of `--option=value`. (Use a space instead of an equals sign.)

Table 8: Options

Option	Description
<code>--class</code>	The FQCN of the class containing the main method of the application. For example, <code>org.apache.spark.examples.SparkPi</code> .
<code>--conf</code>	Spark configuration property in <code>key=value</code> format. For values that contain spaces, surround " <code>key=value</code> " with quotes (as shown).
<code>--deploy-mode</code>	The deployment mode: cluster and client . In cluster mode the driver runs on worker hosts. In client mode, the driver runs locally as an external client. Broadly, cluster mode should be used production jobs, while client mode is more appropriate for interactive and debugging uses, where you want to see your application's output immediately. For affect of the deployment mode when running on YARN, see Deployment Modes on page 227. Default: <code>client</code> .
<code>--driver-cores</code>	Number of cores used by the driver, only in cluster mode (Default: 1).
<code>--driver-memory</code>	The maximum heap size (represented as a JVM string; for example <code>1024m</code> , <code>2g</code> , and so on) to allocate to the driver. Alternatively, you can use the <code>spark.driver.memory</code> configuration parameter.
<code>--jars</code>	Additional JARs to be loaded into the classpath of drivers and executors in cluster mode or into the executor classpath in client mode. These

Option	Description
	JARs can be in the HDFS file system; otherwise they must be available locally on each executor. The path to the JARs on HDFS must be specified as <code>hdfs://nameservice:8020/path/to/jar</code> .
<code>--master</code>	The location to run the application.
<code>--packages</code>	Comma-separated list of Maven coordinates of JARs to include on the driver and executor classpaths. The local Maven repo, then Maven central, and any additional remote repositories specified in <code>--repositories</code> are searched in that order. The format for the coordinates should be <code>groupId:artifactId:version</code> .
<code>--repositories</code>	Comma-separated list of additional remote repositories to search for the Maven coordinates given with <code>--packages</code> .

Table 9: Master Values

Master	Description
<code>local</code>	Run Spark locally with one worker thread (that is, no parallelism).
<code>local[K]</code>	Run Spark locally with <i>K</i> worker threads (ideally, set this to the number of cores on your host).
<code>local[*]</code>	Run Spark locally with as many worker threads as logical cores on your host.
<code>spark://host:port</code>	Run on the Spark Standalone Master on the specified host. The port must be the one your Master is configured to use (7077 by default).
<code>yarn</code>	Run on a YARN cluster. The cluster location is determined by the <code>HADOOP_CONF_DIR</code> or <code>YARN_CONF_DIR</code> variable.

Cluster Overview

MapReduce runs each task in its own process. When a task completes, the process terminates. In Spark, many tasks can run concurrently in an executor, and the executor exists for the lifetime of the Spark application, even when no jobs are running. A cluster manager starts the executor processes.

In this model, tasks can start very quickly and process in-memory data. However, you have less control of resource management. Because the number of executors for an application is typically fixed, and each executor has a fixed allotment of resources, an application uses the same amount of resources for the full duration that it runs.

When running on YARN, you can [dynamically increase and decrease the number of executors](#).

Following are the steps that occur when you submit a Spark application to a cluster:

1. `spark-submit` launches the driver program and invokes the `main` method in the Spark application.
2. The driver program requests resources from the cluster manager to launch executors.
3. The cluster manager launches executors on behalf of the driver program.
4. The driver process runs the user application. Based on the resilient distributed dataset (RDD) actions and transformations in the program, the driver sends tasks to executors.
5. Tasks are run on executor processes to compute and save results.
6. If the driver's `main` method exits or calls `SparkContext.stop`, it terminates the executors and releases resources from the cluster manager.

Table 10: Spark Runtime Summary

Mode	YARN Client Mode	YARN Cluster Mode	Spark Standalone
Driver runs in	Client	ApplicationMaster	Client
Requests resources	ApplicationMaster	ApplicationMaster	Client
Starts executor processes	YARN NodeManager	YARN NodeManager	Spark Worker
Persistent services	YARN ResourceManager and NodeManagers	YARN ResourceManager and NodeManagers	Spark Master and Workers
Supports Spark Shell	Yes	No	Yes

For more information, see [Cluster Mode Overview](#).

Running Spark Applications on YARN

When Spark applications run on YARN, resource management, scheduling, and security are controlled by YARN. You can run an application in cluster mode or client mode.

When running Spark on YARN, each Spark executor runs as a YARN container. Where MapReduce schedules a container and starts a JVM for each task, Spark hosts multiple tasks within the same container. This approach enables several orders of magnitude faster task startup time.

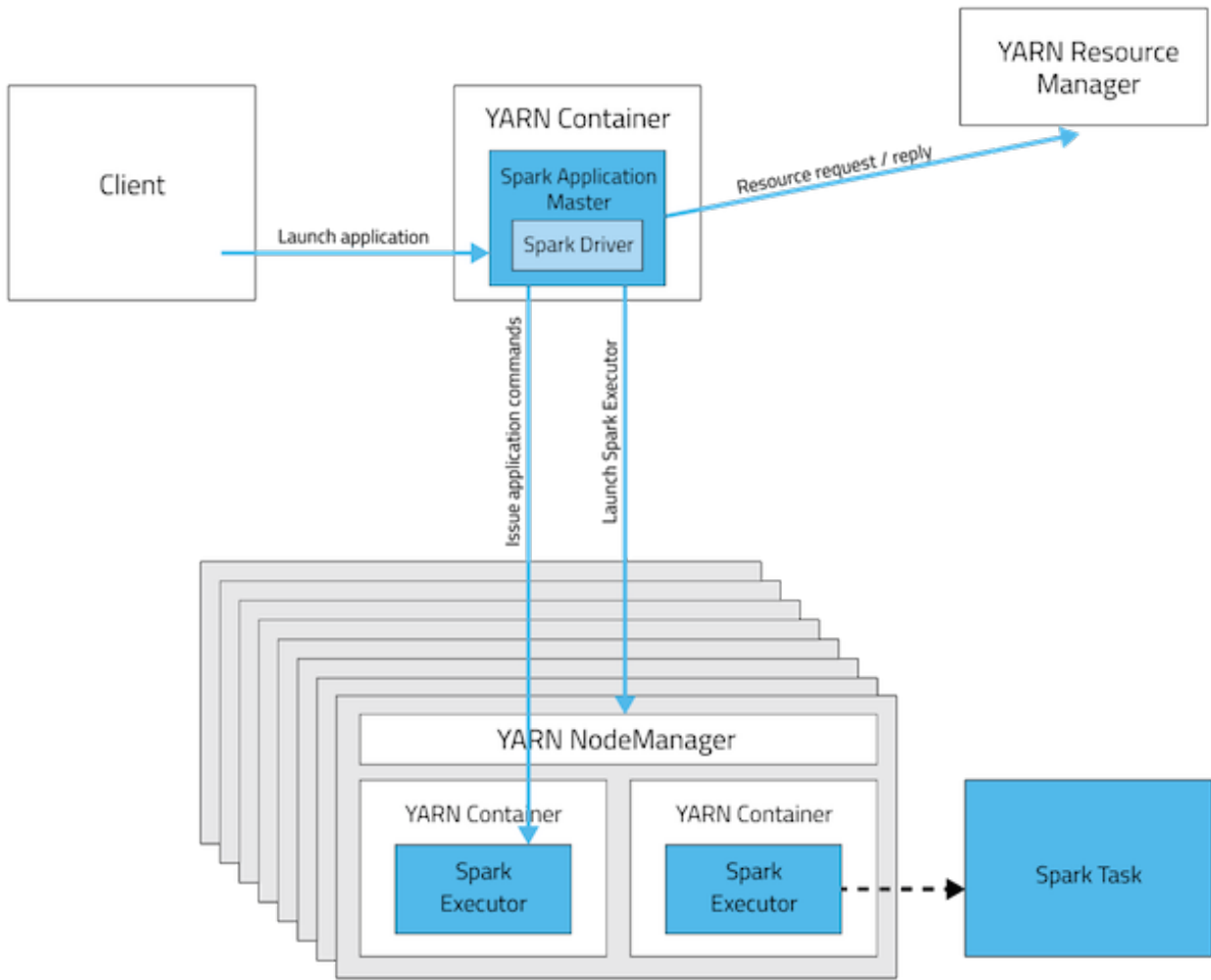
Deployment Modes

In YARN, each application instance has an ApplicationMaster process, which is the first container started for that application. The application is responsible for requesting resources from the ResourceManager. Once the resources are allocated, the application instructs NodeManagers to start containers on its behalf. ApplicationMasters eliminate the need for an active client: the process starting the application can terminate, and coordination continues from a process managed by YARN running on the cluster.

For the option to specify the deployment mode, see [Command-Line Options](#) on page 225.

Cluster Deployment Mode

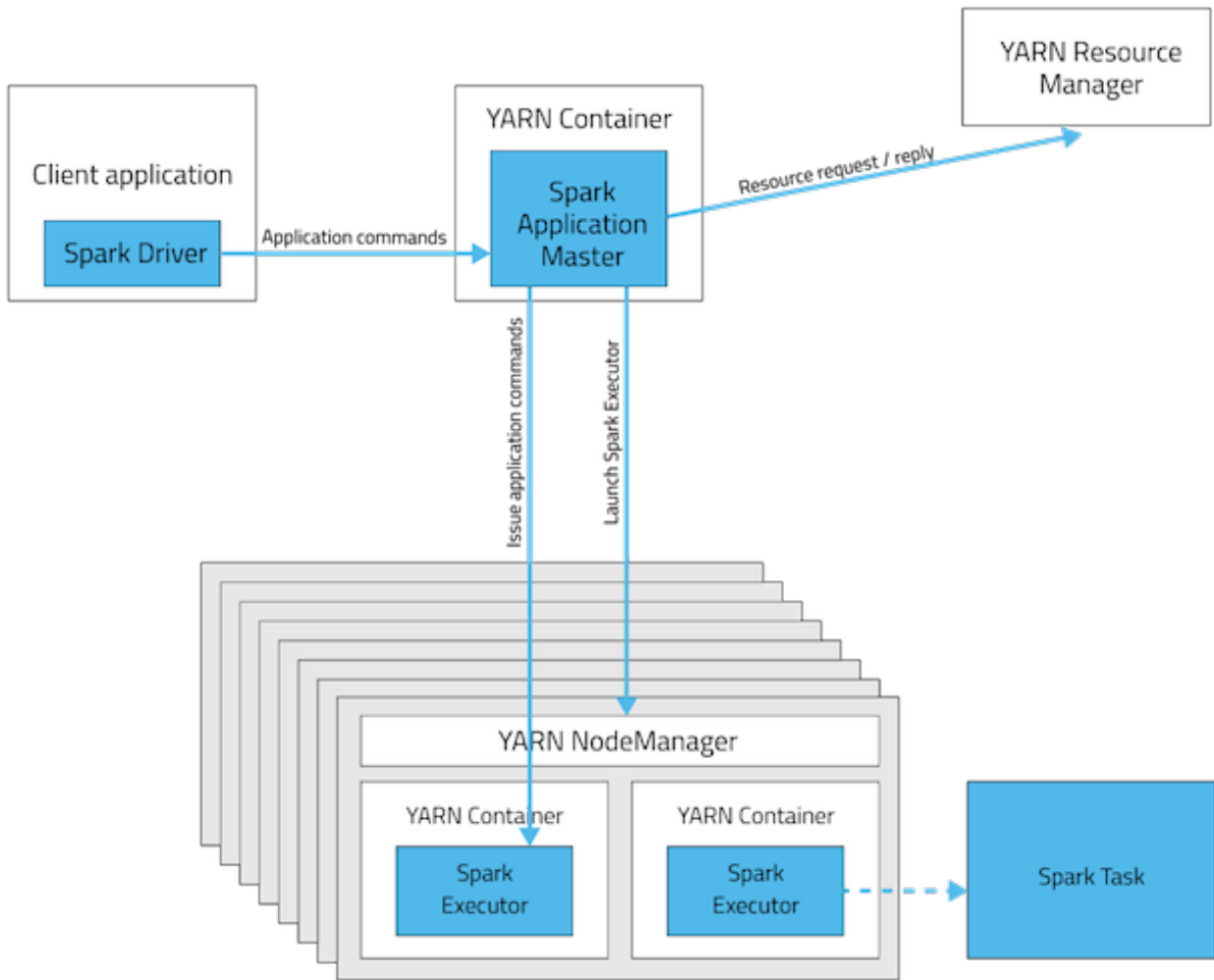
In cluster mode, the driver runs in the ApplicationMaster on a cluster host chosen by YARN. This means that the same process, which runs in a YARN container, is responsible for both driving the application and requesting resources from YARN. The client that launches the application doesn't need to continue running for the entire lifetime of the application.



Cluster mode is not well suited to using Spark interactively. Spark applications that require user input, such as `spark-shell` and `pyspark`, need the Spark driver to run inside the client process that initiates the Spark application.

Client Deployment Mode

In client mode, the driver runs on the host where the job is submitted. The ApplicationMaster is merely present to request executor containers from YARN. The client communicates with those containers to schedule work after they start:



Spark Options

As well as the class implementing the application and the library containing the class, the options applicable to running Spark applications on YARN are:

Option	Description
<code>--executor-cores</code>	Number of processor cores to allocate on each executor.
<code>--executor-memory</code>	The maximum heap size to allocate to each executor. Alternatively, you can use the <code>spark.executor.memory</code> configuration parameter.
<code>--num-executors</code>	The total number of YARN containers to allocate for this application. Alternatively, you can use the <code>spark.executor.instances</code> configuration parameter.
<code>--queue</code>	The YARN queue to submit to. Default: default.

Configuring the Environment

Spark requires the `HADOOP_CONF_DIR` or `YARN_CONF_DIR` environment variable point to the directory containing the client-side configuration files for the cluster. These configurations are used to write to HDFS and connect to the YARN ResourceManager. If you are using a Cloudera Manager deployment, these variables are configured automatically. If you are using an unmanaged deployment, ensure that you set the variables as described in [Running Spark on YARN](#).

Running a Spark Shell Application on YARN

To run the `spark-shell` or `pyspark` client on YARN, use the `--master yarn --deploy-mode client` flags when you start the application.

If you are using a Cloudera Manager deployment, these properties are configured automatically.

Submitting Spark Applications to YARN

To submit an application to YARN, use the `spark-submit` script and specify the `--master yarn` flag. For other `spark-submit` options, see [Table 7: spark-submit Options](#) on page 225.

Monitoring and Debugging Spark Applications

To obtain information about Spark application behavior you can consult YARN logs and the Spark web application UI. These two methods provide complementary information. For information how to view logs created by Spark applications and the Spark web application UI, see [Monitoring Spark Applications](#).

Example: Running SparkPi on YARN

Running SparkPi in YARN Cluster Mode

To run SparkPi in cluster mode:

- CDH 5.2 and lower

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode cluster $SPARK_HOME/examples/lib/spark-examples.jar 10
```

- CDH 5.3 and higher

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode cluster $SPARK_HOME/lib/spark-examples.jar 10
```

The argument passed after the JAR controls how close to Pi the approximation should be.

The command will continue to print out status until the job finishes or you press `control-C`. Terminating the `spark-submit` process in cluster mode does not terminate the Spark application as it does in client mode. To monitor the status of the running application, run `yarn application -list`.

Running SparkPi in YARN Client Mode

To run SparkPi in client mode:

- CDH 5.2 and lower

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client $SPARK_HOME/examples/lib/spark-examples.jar 10
```

- CDH 5.3 and higher

```
spark-submit --class org.apache.spark.examples.SparkPi --master yarn \  
--deploy-mode client $SPARK_HOME/lib/spark-examples.jar 10
```

The argument passed after the JAR controls how close to Pi the approximation should be.

Enabling Dynamic Executor Allocation

Spark can dynamically increase and decrease the number of executors for an application if the resource requirements for the application changes over time. To enable dynamic allocation, set `spark.dynamicAllocation.enabled` to `true`. Specify the minimum number of executors that should be allocated to an application by means of the `spark.dynamicAllocation.minExecutors` parameter, and specify and the maximum number of executors by means of the `spark.dynamicAllocation.maxExecutors` parameter. Set the initial number of executors in the `spark.dynamicAllocation.initialExecutors` parameter. Do not use the `--num-executors` command line

argument or the `spark.executor.instances` parameter; they are incompatible with dynamic allocation. For more information, see [dynamic resource allocation](#).

Optimizing YARN Mode

Unmanaged CDH Deployments

In CDH deployments not managed by Cloudera Manager, Spark copies the Spark assembly JAR file to HDFS each time you run `spark-submit`. You can avoid doing this copy in one of the following ways:

- Set `spark.yarn.jar` to the local path to the assembly JAR:
 - **package installation** - `local:/usr/lib/spark/lib/spark-assembly.jar`
 - **parcel installation** - `local:/opt/cloudera/parcels/CDH/lib/spark/lib/spark-assembly.jar`

This is the default for Cloudera Manager deployments.

- Upload the JAR and configure the JAR location:
 1. Manually upload the Spark assembly JAR file to HDFS:

```
$ hdfs dfs -mkdir -p /user/spark/share/lib
$ hdfs dfs -put $SPARK_HOME/assembly/lib/spark-assembly_*.jar
/user/spark/share/lib/spark-assembly.jar
```

You must manually upload the JAR each time you upgrade Spark to a new minor CDH release.

2. Set `spark.yarn.jar` to the HDFS path:

```
spark.yarn.jar=hdfs://namenode:8020/user/spark/share/lib/spark-assembly.jar
```

Running Spark Applications on Spark Standalone

In CDH 5, Cloudera recommends running Spark applications on a [YARN](#) cluster manager instead of on a Spark Standalone cluster manager, for the following benefits:

- You can dynamically share and centrally configure the same pool of cluster resources among all frameworks that run on YARN.
- You can use [all the features of YARN schedulers](#) for categorizing, isolating, and prioritizing workloads.
- You choose the number of executors to use; in contrast, Spark Standalone requires each application to run an executor on every host in the cluster.
- Spark can run against Kerberos enabled Hadoop clusters and use secure authentication between its processes.

Running a Spark Shell Application on Spark Standalone

To run the `spark-shell` or `pyspark` application on Spark Standalone, use the `--master http://spark_master:spark_master_port` flag when you start the application.

Submitting Spark Applications to Spark Standalone

To submit a Spark application on Spark Standalone, supply the `--master` and `--deploy-mode client` arguments to `spark-submit`.

Example: Running SparkPi on Spark Standalone

- CDH 5.2 and lower:

```
spark-submit --class org.apache.spark.examples.SparkPi --deploy-mode client \
--master http://spark_master:spark_master_port \
$SPARK_HOME/examples/lib/spark-examples.jar 10
```

Managing CDH and Managed Services

- CDH 5.3 and higher:

```
spark-submit --class org.apache.spark.examples.SparkPi --deploy-mode client \  
--master http://spark_master:spark_master_port \  
$SPARK_HOME/lib/spark-examples.jar 10
```

The argument passed after the JAR controls how close to Pi the approximation should be.

Running a Crunch Application with Spark

The blog post [How-to: Run a Simple Apache Spark App in CDH 5](#) provides a tutorial on writing, compiling, and running a Spark application. Taking that article as a starting point, do the following to run a Crunch application with Spark.

1. Add both the `crunch-core` and `crunch-spark` dependencies to your Maven project, along with the other dependencies shown in the blog post.
2. Use the `SparkPipeline` (`org.apache.crunch.impl.spark.SparkPipeline`) where you would have used the `MRPipeline` instance in the declaration of your Crunch pipeline. The `SparkPipeline` will need either a String that contains the connection string for the Spark master (`local` for local mode, `yarn-client` for YARN) or an actual `JavaSparkContext` instance.
3. Update the `SPARK_SUBMIT_CLASSPATH`:

```
export  
SPARK_SUBMIT_CLASSPATH=./commons-codec-1.4.jar:$SPARK_HOME/assembly/lib/*:./myapp-jar-with-dependencies.jar
```



Important: The `commons-codec-1.4` dependency must come before the `SPARK_HOME` dependencies.

4. Now you can start the pipeline using your Crunch application `jar-with-dependencies` file using the `spark-submit` script, just as you would for a regular Spark pipeline.

Managing the Sqoop 1 Client

The Sqoop 1 client allows you to create a Sqoop 1 [gateway](#) and deploy the client configuration.

Installing JDBC Drivers

Sqoop 1 does not ship with third-party JDBC drivers; you must download them separately. For information on downloading and saving the drivers, see [\(CDH 4\) Installing JDBC Drivers](#) and [\(CDH 5\) Installing JDBC Drivers](#). Ensure that you do not save JARs in the CDH parcel directory `/opt/cloudera/parcels/CDH`, because this directory is overwritten when you upgrade CDH.

Adding the Sqoop 1 Client

Minimum Required Role: [Full Administrator](#)

The Sqoop 1 client packages are installed by the Installation wizard. However, the client configuration is not deployed. To create a Sqoop 1 gateway and deploy the client configuration:

1. On the **Home > Status** tab, click



to the right of the cluster name and select **Add a Service**. A list of service types display. You can add one type of service at a time.

2. Select the **Sqoop 1 Client** service and click **Continue**.
3. Select the radio button next to the services on which the new service should depend. All services must depend on the *same* ZooKeeper service. Click **Continue**.
4. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. Click **Continue**. The client configuration deployment command runs.
6. Click **Continue** and click **Finish**.

Managing Sqoop 2

Cloudera Manager installs the Sqoop 2 service as part of the CDH installation.



Note: Sqoop 2 is being deprecated. Cloudera recommends you use Sqoop 1.

You can elect to have the service created and started as part of the Installation wizard. If you elect not to create the service using the Installation wizard, you can use the **Add Service** wizard to perform the installation. The wizard will automatically configure and start the dependent services and the Sqoop 2 service. See [Adding a Service](#) on page 36 for instructions.

Installing JDBC Drivers

The Sqoop 2 service does not ship with third-party JDBC drivers; you must download them separately. For information on downloading and saving the drivers, see [\(CDH 4\) Configuring Sqoop 2](#) and [\(CDH 5\) Configuring Sqoop 2](#). Ensure that you do not save JARs in the CDH parcel directory `/opt/cloudera/parcels/CDH`, because this directory is overwritten when you upgrade CDH.

Managing ZooKeeper

Minimum Required Role: [Full Administrator](#)

When adding the ZooKeeper service, the **Add Service** wizard automatically initializes the data directories. If you quit the **Add Service** wizard or it does not finish successfully, you can initialize the directories outside the wizard by doing these steps:

1. Go to the ZooKeeper service.
2. Select **Actions > Initialize**.
3. Click **Initialize** again to confirm.



Note: If the data directories are not initialized, the ZooKeeper servers cannot be started.

In a production environment, you should deploy ZooKeeper as an ensemble with an odd number of servers. As long as a majority of the servers in the ensemble are available, the ZooKeeper service will be available. The minimum

Managing CDH and Managed Services

recommended ensemble size is three ZooKeeper servers, and Cloudera recommends that each server run on a separate machine. In addition, the ZooKeeper server process should have its own dedicated disk storage if possible.

Using Multiple ZooKeeper Services

Cloudera Manager requires dependent services within CDH to use the same ZooKeeper service. If you configure dependent CDH services to use different ZooKeeper services, Cloudera Manager reports the following error:

```
com.cloudera.cmf.command.CmdExecException: java.lang.RuntimeException:
java.lang.IllegalStateException: Assumption violated:
getAllDependencies returned multiple distinct services of the same type
at SeqFlowCmd.java line 120
in com.cloudera.cmf.command.flow.SeqFlowCmd run()
```

CDH services that are not dependent can use different ZooKeeper services. For example, Kafka does not depend on any services other than ZooKeeper. You might have one ZooKeeper service for Kafka, and one ZooKeeper service for the rest of your CDH services.

Configuring Services to Use the GPL Extras Parcel

After you [install the GPL Extras parcel](#), reconfigure and restart services that need to use LZO functionality. Any service that does not require the use of LZO need not be configured.

HDFS

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Search for the `io.compression.codecs` property.
4. In the **Compression Codecs** property, click in the field, then click the + sign to open a new value field.
5. Add the following two codecs:
 - `com.hadoop.compression.lzo.LzoCodec`
 - `com.hadoop.compression.lzo.LzopCodec`
6. Save your configuration changes.
7. Restart HDFS.
8. Redeploy the HDFS client configuration.

Oozie

1. Go to `/var/lib/oozie` on each Oozie server and even if the LZO JAR is present, symlink the Hadoop LZO JAR:
 - **CDH 5** - `/opt/cloudera/parcels/GPLEXTRAS/lib/hadoop/lib/hadoop-lzo.jar`
 - **CDH 4** - `/opt/cloudera/parcels/HADOOP_LZO/lib/hadoop/lib/hadoop-lzo.jar`
2. Restart Oozie.

HBase

Restart HBase.

Impala

Restart Impala.

Hive

Restart the Hive server.

Sqoop 2

1. Add the following entries to the **Sqoop Service Environment Advanced Configuration Snippet**:

- `HADOOP_CLASSPATH=$HADOOP_CLASSPATH:/opt/cloudera/parcels/GPLEXTRAS/lib/hadoop/lib/*`
- `JAVA_LIBRARY_PATH=$JAVA_LIBRARY_PATH:/opt/cloudera/parcels/GPLEXTRAS/lib/hadoop/lib/native`

2. Restart the Sqoop service.

Resource Management

Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources. The goals of resource management features are to:

- Guarantee completion in a reasonable time frame for critical workloads
- Support reasonable cluster scheduling between groups of users based on fair allocation of resources per group
- Prevent users from depriving other users access to the cluster

Schedulers

A scheduler is responsible for deciding which tasks get to run and where and when to run them.

The MapReduce and YARN computation frameworks support the following schedulers:

- **FIFO** - Allocates resources based on arrival time.
- **Fair** - Allocates resources to weighted pools, with fair sharing within each pool.
 - [CDH 4 Fair Scheduler](#)
 - [CDH 5 Fair Scheduler](#)
- **Capacity** - Allocates resources to pools, with FIFO scheduling within each pool.
 - [CDH 4 Capacity Scheduler](#)
 - [CDH 5 Capacity Scheduler](#)

The scheduler defaults for MapReduce and YARN are:

- **MapReduce** - Cloudera Manager, CDH 5, and CDH 4 set the default to FIFO. FIFO is set as the default for backward-compatibility purposes, but Cloudera recommends Fair Scheduler because Impala and Llama are optimized for it. Capacity Scheduler is also available.

If you are running CDH 4, you can specify how MapReduce jobs share resources by [configuring the MapReduce scheduler](#).

- **YARN** - Cloudera Manager, CDH 5, and CDH 4 set the default to Fair Scheduler. Cloudera recommends Fair Scheduler because Impala and Llama are optimized for it. FIFO and Capacity Scheduler are also available.

In YARN, the scheduler is responsible for allocating resources to the various running applications subject to familiar constraints of capacities, queues, and so on. The scheduler performs its scheduling function based the resource requirements of the applications; it does so based on the abstract notion of a resource *container* that incorporates elements such as memory, CPU, disk, network, and so on.

The YARN scheduler has a pluggable policy plug-in, which is responsible for partitioning the cluster resources among the various queues, applications, and so on.

If you are running CDH 5, you can specify how YARN applications share resources by manually [configuring the YARN scheduler](#). Alternatively you can use Cloudera Manager [dynamic allocation](#) features to manage the scheduler configuration.

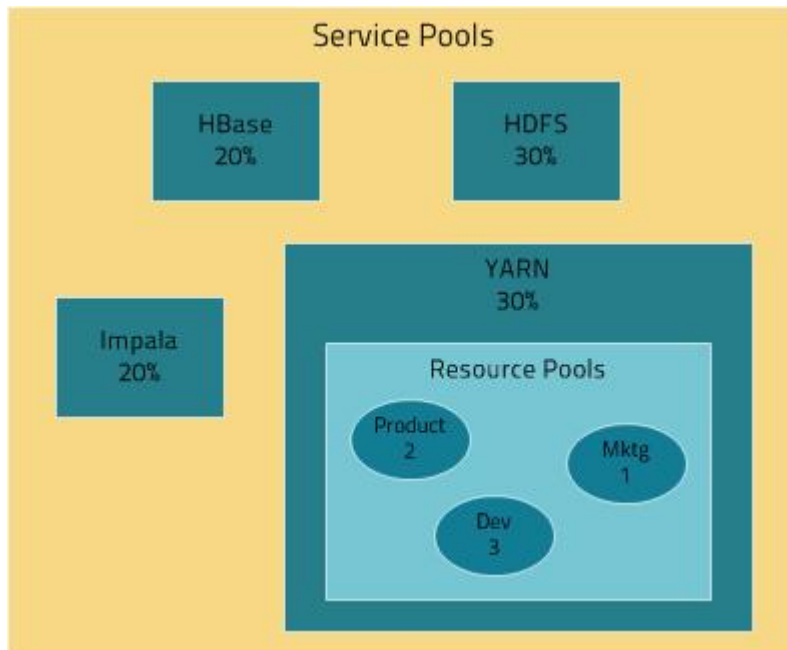
Cloudera Manager Resource Management Features

Cloudera Manager provides the following features to assist you with allocating cluster resources to services.

Static Allocation

Cloudera Manager 4 introduced the ability to partition resources across HBase, HDFS, Impala, MapReduce, and YARN services by allowing you to set configuration properties that were enforced by Linux control groups (Linux cgroups). With Cloudera Manager 5, the ability to statically allocate resources using cgroups is configurable through a single *static service pool wizard*. You allocate services a percentage of total resources and the wizard configures the cgroups.

For example, the following figure illustrates static pools for HBase, HDFS, Impala, and YARN services that are respectively assigned 20%, 30%, 20%, and 30% of cluster resources.



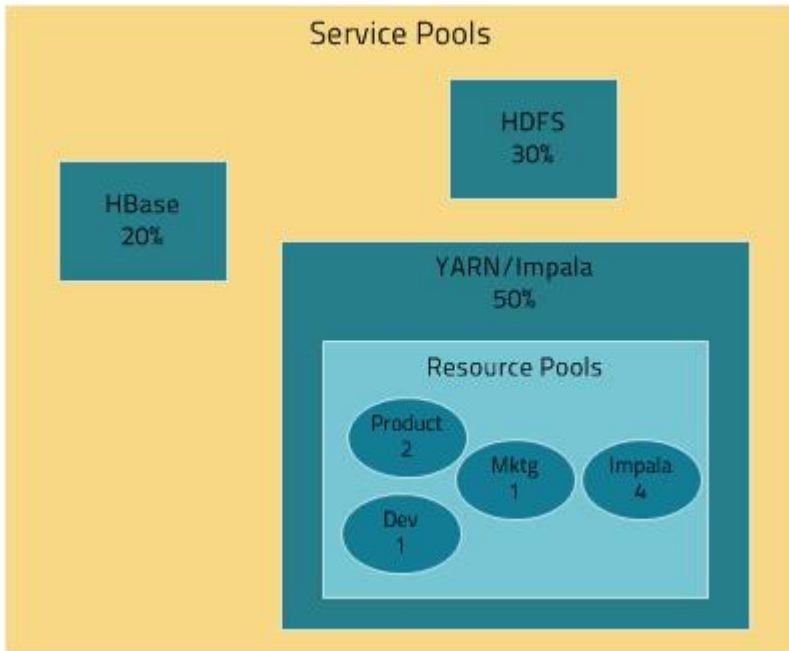
Dynamic Allocation

Cloudera Manager allows you to manage mechanisms for dynamically apportioning resources statically allocated to YARN and Impala using *dynamic resource pools*.

Depending on the version of CDH you are using, dynamic resource pools in Cloudera Manager support the following resource management (RM) scenarios:

- **(CDH 5) YARN Independent RM** - YARN manages the virtual cores, memory, running applications, and scheduling policy for each pool. In the preceding diagram, three dynamic resource pools - Dev, Product, and Mktg with weights 3, 2, and 1 respectively - are defined for YARN. If an application starts and is assigned to the Product pool, and other applications are using the Dev and Mktg pools, the Product resource pool will receive $30\% \times 2/6$ (or 10%) of the total cluster resources. If there are no applications using the Dev and Mktg pools, the YARN Product pool will be allocated 30% of the cluster resources.
- **(CDH 5) YARN and Impala Integrated RM** - YARN manages memory for pools running Impala queries; Impala limits the number of running and queued queries in each pool. In the YARN and Impala integrated RM scenario, Impala services can reserve resources through YARN, effectively sharing the static YARN service pool and resource pools with YARN applications. The integrated resource management scenario, where both YARN and Impala use the YARN resource management framework, require the [Impala Llama](#) role.

In the following figure, the YARN and Impala services have a 50% static share which is subdivided among the original resource pools with an additional resource pool designated for the Impala service. If YARN applications are using all the original pools, and Impala uses its designated resource pool, Impala queries will have the same resource allocation $50\% \times 4/8 = 25\%$ as in the first scenario. However, when YARN applications are not using the original pools, Impala queries will have access to 50% of the cluster resources.



Note:

When using YARN with Impala, Cloudera recommends using the static partitioning technique (through a static service pool) rather than the combination of YARN and Llama. YARN is a central, synchronous scheduler and thus introduces higher latency and variance which is better suited for batch processing than for interactive workloads like Impala (especially with higher concurrency). Currently, YARN allocates memory throughout the query, making it hard to reason about out-of-memory and timeout conditions.

- **(CDH 5) YARN and Impala Independent RM** - YARN manages the virtual cores, memory, running applications, and scheduling policy for each pool; Impala manages memory for pools running queries and limits the number of running and queued queries in each pool.
- **(CDH 5 and CDH 4) Impala Independent RM** - Impala manages memory for pools running queries and limits the number of running and queued queries in each pool.

The scenarios where YARN manages resources, whether for independent RM or integrated RM, map to the YARN scheduler configuration. The scenarios where Impala independently manages resources employ the Impala [admission control](#) feature.

To submit a YARN application to a specific resource pool, specify the `mapreduce.job.queueName` property. The YARN application's queue property is mapped to a resource pool. To submit an Impala query to a specific resource pool, specify the `REQUEST_POOL` option.

Managing Resources with Cloudera Manager

Linux Control Groups

Minimum Required Role: [Full Administrator](#)

Cloudera Manager supports the Linux control groups (cgroups) kernel feature. With cgroups, administrators can impose per-resource restrictions and limits on services and roles. This provides the ability to allocate resources using cgroups to enable isolation of compute frameworks from one another. Resource allocation is implemented by setting properties for the services and roles.

Linux Distribution Support

Cgroups are a feature of the Linux kernel, and as such, support depends on the host's Linux distribution and version. As shown in the following table, Linux cgroups don't exist on RHEL 5. However, all of the other OS platforms supported by Cloudera Manager support cgroups.

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
Red Hat Enterprise Linux (or CentOS) 5				
Red Hat Enterprise Linux (or CentOS) 6	■	■	■	■
SUSE Linux Enterprise Server 11	■	■	■	■
Ubuntu 10.04 LTS	■		■	■
Ubuntu 12.04 LTS	■	■	■	■
Ubuntu 12.04 LTS	■	■	■	■
Debian 6.0	■			
Debian 7.0	■			
Debian 7.1	■			

If a distribution lacks support for a given parameter, changes to the parameter have no effect.

The exact level of support can be found in the Cloudera Manager Agent log file, shortly after the Agent has started. See [Viewing Cloudera Manager Server and Agent Logs](#) to find the Agent log. In the log file, look for an entry like this:

```
Found cgroups capabilities: {'has_memory': True, 'default_memory_limit_in_bytes':
9223372036854775807,
'writable_cgroup_dot_procs': True, 'has_cpu': True,
'default_blkio_weight': 1000, 'default_cpu_shares': 1024, 'has_blkio': True}
```

The `has_memory` and similar entries correspond directly to support for the CPU, I/O, and memory parameters.

Further Reading

- <http://www.kernel.org/doc/Documentation/cgroups/cgroups.txt>
- <http://www.kernel.org/doc/Documentation/cgroups/blkio-controller.txt>
- <http://www.kernel.org/doc/Documentation/cgroups/memory.txt>
- https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Resource_Management_Guide/index.html

Resource Management with Control Groups

To use cgroups, you must also enable cgroup-based resource management under the **Host > Resource Management** configuration properties. However, if you configure [static service pools](#), this property will be set as part of that process.

Enabling Resource Management

Cgroups-based resource management can be enabled for all hosts, or on a per-host basis.

1. If you have upgraded from a version of Cloudera Manager older than Cloudera Manager 4.5, restart every Cloudera Manager Agent before using cgroups-based resource management:
 - a. Stop all services, including the Cloudera Management Service.
 - b. On each cluster host, run as root:

```
$ sudo service cloudera-scm-agent hard_restart
```

- c. Start all services.
2. Click the **Hosts** tab.
3. Optionally click the link for the host where you want to enable cgroups.
4. Click the **Configuration** tab.
5. Select **Category > Resource Management**.
6. Select **Enable Cgroup-based Resource Management**.
7. Restart all roles on the host or hosts.

Limitations

- Role group and role instance override cgroup-based resource management parameters must be saved one at a time. Otherwise some of the changes that should be reflected dynamically will be ignored.
- The role group abstraction is an imperfect fit for resource management parameters, where the goal is often to take a numeric value for a host resource and distribute it amongst running roles. The role group represents a "horizontal" slice: the same role across a set of hosts. However, the cluster is often viewed in terms of "vertical" slices, each being a combination of worker roles (such as TaskTracker, DataNode, Region Server, Impala Daemon, and so on). Nothing in Cloudera Manager guarantees that these disparate horizontal slices are "aligned" (meaning, that the role assignment is identical across hosts). If they are unaligned, some of the role group values will be incorrect on unaligned hosts. For example a host whose role groups have been configured with memory limits but that's missing a role will probably have unassigned memory.

Configuring Resource Parameters

After enabling cgroups, you can restrict and limit the resource consumption of roles (or role groups) on a per-resource basis. All of these parameters can be found in the Cloudera Manager Admin Console, under the Resource Management category:

- **CPU Shares** - The more CPU shares given to a role, the larger its share of the CPU when under contention. Until processes on the host (including both roles managed by Cloudera Manager and other system processes) are contending for all of the CPUs, this will have no effect. When there is contention, those processes with higher CPU shares will be given more CPU time. The effect is linear: a process with 4 CPU shares will be given roughly twice as much CPU time as a process with 2 CPU shares.
Updates to this parameter will be dynamically reflected in the running role.
- **I/O Weight** - The greater the I/O weight, the higher priority will be given to I/O requests made by the role when I/O is under contention (either by roles managed by Cloudera Manager or by other system processes). This only affects read requests; write requests remain unprioritized.
Updates to this parameter will be dynamically reflected in the running role.
- **Memory Soft Limit** - When the limit is reached, the kernel will reclaim pages charged to the process if and only if the host is facing memory pressure. If reclaiming fails, the kernel may kill the process. Both anonymous as well as page cache pages contribute to the limit.
After updating this parameter, the role must be restarted before changes take effect.
- **Memory Hard Limit** - When a role's resident set size (RSS) exceeds the value of this parameter, the kernel will swap out some of the role's memory. If it's unable to do so, it will kill the process. Note that the kernel measures memory consumption in a manner that doesn't necessarily match what the `top` or `ps` report for RSS, so expect that this limit is a rough approximation.
After updating this parameter, the role must be restarted before changes take effect.

Example: Protecting Production MapReduce Jobs from Impala Queries

Suppose you have MapReduce deployed in production and want to roll out Impala without affecting production MapReduce jobs. For simplicity, we will make the following assumptions:

- The cluster is using homogenous hardware

- Each worker host has two cores
- Each worker host has 8 GB of RAM
- Each worker host is running a DataNode, TaskTracker, and an Impala Daemon
- Each role type is in a single role group
- Cgroups-based resource management has been enabled on all hosts

Action	Procedure
CPU	<ol style="list-style-type: none"> 1. Leave DataNode and TaskTracker role group CPU shares at 1024. 2. Set Impala Daemon role group's CPU shares to 256. 3. The TaskTracker role group should be configured with a Maximum Number of Simultaneous Map Tasks of 2 and a Maximum Number of Simultaneous Reduce Tasks of 1. This yields an upper bound of three MapReduce tasks at any given time; this is an important detail for memory sizing.
Memory	<ol style="list-style-type: none"> 1. Set Impala Daemon role group memory limit to 1024 MB. 2. Leave DataNode maximum Java heap size at 1 GB. 3. Leave TaskTracker maximum Java heap size at 1 GB. 4. Leave MapReduce Child Java Maximum Heap Size for Gateway at 1 GB. 5. Leave cgroups hard memory limits alone. We'll rely on "cooperative" memory limits exclusively, as they yield a nicer user experience than the cgroups-based hard memory limits.
I/O	<ol style="list-style-type: none"> 1. Leave DataNode and TaskTracker role group I/O weight at 500. 2. Impala Daemon role group I/O weight is set to 125.

When you're done with configuration, restart all services for these changes to take effect.

The results are:

1. When MapReduce jobs are running, all Impala queries together will consume up to a fifth of the cluster's CPU resources.
2. Individual Impala Daemons won't consume more than 1 GB of RAM. If this figure is exceeded, new queries will be cancelled.
3. DataNodes and TaskTrackers can consume up to 1 GB of RAM each.
4. We expect up to 3 MapReduce tasks at a given time, each with a maximum heap size of 1 GB of RAM. That's up to 3 GB for MapReduce tasks.
5. The remainder of each host's available RAM (6 GB) is reserved for other host processes.
6. When MapReduce jobs are running, read requests issued by Impala queries will receive a fifth of the priority of either HDFS read requests or MapReduce read requests.

Static Service Pools

Static service pools isolate the services in your cluster from one another, so that load on one service has a bounded impact on other services. Services are allocated a static percentage of total resources—CPU, memory, and I/O weight—which are not shared with other services. When you configure static service pools, Cloudera Manager computes recommended memory, CPU, and I/O configurations for the worker roles of the services that correspond to the percentage assigned to each service. Static service pools are implemented per role group within a cluster, using [Linux control groups \(cgroups\)](#) and cooperative memory limits (for example, Java maximum heap sizes). Static service pools can be used to control access to resources by HBase, HDFS, Impala, MapReduce, Solr, Spark, YARN, and [add-on](#) services. Static service pools are not enabled by default.



Note:

- I/O allocation only works when [short-circuit reads](#) are enabled.
- I/O allocation does not handle write side I/O because cgroups in the Linux kernel do not currently support buffered writes.

Viewing Static Service Pool Status

Select **Clusters** > **Cluster name** > **Resource Management** > **Static Service Pools**. If the cluster has a YARN service, the Static Service Pools Status tab displays and shows whether resource management is enabled for the cluster, and the currently configured service pools.

See [Monitoring Static Service Pools](#) for more information.

Enabling and Configuring Static Service Pools

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

1. Select **Clusters** > **Cluster name** > **Resource Management** > **Static Service Pools**.
2. Click the **Configuration** tab. The **Step 1 of 4: Basic Allocation Setup** page displays. In each field in the basic allocation table, enter the percentage of resources to give to each service. The total must add up to 100%. In CDH 5 clusters, if you [enable integrated resource management](#), the Impala service shares the YARN service pool, rather than use its own static pool. In this case, you cannot specify a percentage for the Impala service. Click **Continue** to proceed.
3. **Step 2: Review Changes** - The allocation of resources for each resource type and role displays with the new values as well as the values previously in effect. The values for each role are set by role group; if there is more than one role group for a given role type (for example, for RegionServers or DataNodes) then resources will be allocated separately for the hosts in each role group. Take note of changed settings. If you have previously customized these settings, check these over carefully.
 - Click the ➤ to the right of each percentage to display the allocations for a single service. Click ➤ to the right of the Total (100%) to view all the allocations in a single page.
 - Click the **Back** button to go to the previous page and change your allocations.

When you are satisfied with the allocations, click **Continue**.

4. **Step 3 of 4: Restart Services** - To apply the new allocation percentages, click **Restart Now** to restart the cluster. To skip this step, click **Restart Later**. If HDFS High Availability is enabled, you will have the option to choose a [rolling restart](#).
5. **Step 4 of 4: Progress** displays the status of the restart commands. Click **Finished** after the restart commands complete.

Disabling Static Service Pools

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

To disable static service pools, disable cgroup-based resource management for all hosts in all clusters:

1. In the main navigation bar, click **Hosts**.
2. Click the **Configuration** tab.
3. Select **Scope** > **Resource Management**.
4. Deselect the **Enable Cgroup-based Resource Management** property.
5. Click **Save Changes**.
6. Restart all services.

Static resource management is disabled, but the percentages you set when you configured the pools, and all the changed settings (for example, heap sizes), *are retained* by the services. The percentages and settings will also be used when you re-enable static service pools. If you want to revert to the settings you had before static service pools were enabled, follow the procedures in [Viewing and Reverting Configuration Changes](#) on page 30.

Dynamic Resource Pools

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

A **dynamic resource pool** is a named configuration of resources and a policy for scheduling the resources among YARN applications and Impala queries running in the pool. Dynamic resource pools allow you to schedule and allocate resources to YARN applications and Impala queries based on a user's access to specific pools and the resources available to those pools. If a pool's allocation is not in use it can be given to other pools. Otherwise, a pool receives a share of

resources in accordance with the pool's weight. Dynamic resource pools have ACLs that restrict who can submit work to and administer them.

A **configuration set** defines the allocation of resources across pools that may be active at a given time. For example, you can define "weekday" and "weekend" configuration sets, which define different resource pool configurations for different days of the week.

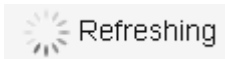
A **scheduling rule** defines when a configuration set is active. The configuration set is updated in affected services every hour.

Resource pools can be nested, with sub-pools restricted by the settings of their parent pool.

The resources available for sharing are subject to the allocations made for each service if [static service pools](#) (cgroups) are being enforced. For example, if the static pool for YARN is 75% of the total cluster resources, then resource pools will use only that 75% of resources.

Managing Dynamic Resource Pools

After you create or edit a resource pool,



displays while the settings are propagated to the service configuration files. You can also manually [refresh the files](#).

Viewing Dynamic Resource Pool Configuration

Depending on which resource management scenario described in [Cloudera Manager Resource Management Features](#) on page 236 is in effect, the dynamic resource pool configuration overview displays the following information:

- **YARN Independent RM** - Weight, Virtual Cores, Min and Max Memory, Max Running Apps, and Scheduling Policy
- **YARN and Impala Integrated RM**
 - **YARN** - Weight, Virtual Cores, Min and Max Memory, Max Running Apps, and Scheduling Policy
 - **Impala** - Max Running Queries and Max Queued Queries
- **YARN and Impala Independent RM**
 - **YARN** - Weight, Virtual Cores, Min and Max Memory, Max Running Apps, and Scheduling Policy
 - **Impala** - Max Memory, Max Running Queries, and Max Queued Queries
- **Impala Independent RM** - Max Memory, Max Running Queries, and Max Queued Queries

To view dynamic resource pool configuration:

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.

Enabling and Disabling Dynamic Resource Pools for Impala

By default dynamic resource pools for Impala are disabled. If dynamic resource pools are disabled, the Impala section will not appear in the Dynamic Resource Pools tab or in the resource pool dialogs within that page. To modify the Impala dynamic resource pool setting:

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Category > Admission Control**.
4. Select or deselect the **Enable Dynamic Resource Pools** checkbox.
5. Click **Save Changes** to commit the changes.
6. Restart the Impala service.


Creating a Dynamic Resource Pool

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click **Add Resource Pool**. The Add dialog box displays showing the **General** tab.
4. Specify a name and resource limits for the pool:
 - In the **Resource Pool Name** field, specify the pool name. Enter a unique name containing only alphanumeric characters. If referencing a user or group name that contains a ".", replace the "." with "_dot_".
 - Specify the policy for scheduling resources among applications running in the pool:
 - **Dominant Resource Fairness (DRF) (default)** - An extension of fair scheduling for more than one resource—it determines resource shares (CPU, memory) for a job separately based on the availability of those resources and the needs of the job.
 - **Fair Scheduler (FAIR)** - Determines resource shares based on memory.
 - **First-In, First-Out (FIFO)** - Determines resource shares based on when the job was added.
 - If you have enabled Fair Scheduler preemption, optionally set a preemption timeout to specify how long a job in this pool must wait before it can preempt resources from jobs in other pools. To enable preemption, click the [Fair Scheduler Preemption](#) link or follow the procedure in [Enabling Preemption](#) on page 246.
5. Do one or more of the following:
 - Click the **YARN** tab.
 1. Click a [configuration set](#).
 2. Specify a weight that indicates that pool's share of resources relative to other pools, minimum and maximums for virtual cores and memory, and a limit on the number of applications that can run simultaneously in the pool.
 - Click the **Impala** tab.
 1. Click a [configuration set](#).
 2. Specify the maximum number of concurrently running and queued queries in the pool.
6. If you have [enabled ACLs and specified users or groups](#), optionally click the **Submission** and **Administration Access Control** tabs to specify which users and groups can submit applications and which users can view all and kill applications. The default is that anyone can submit, view all, and kill applications. To restrict either of these permissions, select the **Allow these users and groups** radio button and provide a comma-delimited list of users and groups in the Users and Groups fields respectively. Click **OK**.

Adding Sub-Pools

Pools can be nested as sub-pools. They share among their siblings the resources of the parent pool. Each sub-pool can have its own resource restrictions; if those restrictions fall within the configuration of the parent pool, then the limits for the sub-pool take effect. If the limits for the sub-pool exceed those of the parent, then the parent limits take effect.

Once you create sub-pools, jobs cannot be submitted to the parent pool; they must be submitted to a sub-pool.

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click  at the right of a resource pool row and select **Add Sub Pool**. Configure sub-pool properties.
4. Click **OK**.

Configuring Default Scheduler Properties

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click the **Default Settings** button.
4. Specify the default scheduling policy, maximum applications, and preemption timeout properties.
5. Click **OK**.

Editing Dynamic Resource Pools

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click **Edit** at the right of a resource pool row. Edit the properties and click **OK**.
4. If you have [enabled ACLs and specified users or groups](#), optionally click the **Submission** and **Administration Access Control** tabs to specify which users and groups can submit applications and which users can view all and kill applications. The default is that anyone can submit, view all, and kill applications. To restrict either of these permissions, select the **Allow these users and groups** radio button and provide a comma-delimited list of users and groups in the Users and Groups fields respectively. Click **OK**.

Refreshing Dynamic Resource Pool Configuration Files

After updating resource pool settings, you can refresh service configuration files as follows:

1. On the Home page Status tab, select **Clusters > Cluster name > Refresh Dynamic Resource Pools**.

YARN Pool Status and Configuration Options

Viewing Dynamic Resource Pool Status

Select **Clusters > ClusterName > Dynamic Resource Pools**. The Status tab displays the YARN resource pools currently in use for the cluster. See [Monitoring Dynamic Resource Pools](#) for more information.

Setting User Limits



Pool properties determine the maximum number of applications that can run in a pool. To limit the number of applications specific users can run at the same time in a pool:

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**.
2. Click the **Configuration** tab.
3. Click the **User Limits** tab. The table displays a list of users and the maximum number of jobs each user can submit.
4. Click **Add User Limit**.
5. Specify a username. Enter a unique name containing only alphanumeric characters. If referencing a user or group name that contains a ".", replace the "." with "_dot_".
6. Specify the maximum number of running applications.
7. Click **OK**.

Enabling ACLs



To specify whether ACLs are checked:

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**.
2. Click the **Configuration** tab.
3. Click **Other Settings**.

4. In the **Enable ResourceManager ACLs** property, click . The YARN service configuration page displays.
5. Select the checkbox.
6. Click **Save Changes** to commit the changes.
7. Click  to invoke the cluster restart wizard.
8. Click **Restart Cluster**.
9. Click **Restart Now**.
10. Click **Finish**.



Configuring ACLs

To configure which users and groups can submit and kill YARN applications in any resource pool:

1. [Enable ACLs](#).
2. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**.
3. Click the **Configuration** tab.
4. Click **Other Settings**.
5. In the **Admin ACL** property, click . The YARN service configuration page displays.
6. Specify which users and groups can submit and kill applications.
7. Click **Save Changes** to commit the changes.
8. Click  to invoke the cluster restart wizard.
9. Click **Restart Cluster**.
10. Click **Restart Now**.
11. Click **Finish**.

Enabling Preemption

You can enable the Fair Scheduler to preempt applications in other pools if a pool's minimum share is not met for some period of time. When you [create a pool](#) you can specify how long a pool must wait before other applications are preempted.

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**.
2. Click the **Configuration** tab.
3. Click the **User Limits** tab. The table shows you a list of users and the maximum number of jobs each user can submit.
4. Click **Other Settings**.
5. In the **Fair Scheduler Preemption**, click . The YARN service configuration page displays.
6. Select the checkbox.
7. Click **Save Changes** to commit the changes.
8. Click  to invoke the cluster restart wizard.
9. Click **Restart Cluster**.
10. Click **Restart Now**.
11. Click **Finish**.


Placement Rules

Cloudera Manager provides many options for determining how YARN applications and Impala queries are placed in resource pools. You can specify basic rules that place applications and queries in pools based on runtime configurations or the name of the user running the application or query or select an advanced option that allows you to specify a set of ordered rules for placing applications and queries in pools.

To submit a YARN application to a specific resource pool, specify the `mapreduce.job.queueName` property. The YARN application's queue property is mapped to a resource pool. To submit an Impala query to a specific resource pool, specify the `REQUEST_POOL` option.

Enabling and Disabling Undeclared Pools

If you do not specify a pool with a job or query property, by default YARN and Impala create a pool "on-the-fly" with the name of the user that submitted the request and assigns it to that resource pool. For YARN, you can change this behavior so that the **default** pool is used instead:


1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click the **Placement Rules** tab.
4. Click **Basic** radio button.
5. Click the **Allow Undeclared Pools** property.
6. Select or deselect the **Allow Undeclared Pools** checkbox.
7. Click **Save Changes** to commit the changes.
8. Click  to invoke the cluster restart wizard.
9. Click **Restart Cluster**.
10. Click **Restart Now**.
11. Click **Finish**.



Note: YARN and Impala pools created "on-the-fly" are deleted when you restart the YARN and Impala services.


Enabling and Disabling the Default Pool

If an application specifies a pool that has not been explicitly configured or is assigned to a pool with the name of user according to the **Fair Scheduler User As Default Queue** property, by default YARN creates the pool at runtime with default settings. To change the behavior so that under these circumstances the **default** pool is used instead:

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click the **Placement Rules** tab.
4. Click **Basic** radio button.
5. Click the **Fair Scheduler User As Default Queue** property.
6. Select or deselect the checkbox.
7. Click **Save Changes** to commit the changes.
8. Click  to invoke the cluster restart wizard.
9. Click **Restart Cluster**.
10. Click **Restart Now**.
11. Click **Finish**.

Specifying Advanced Placement Rules and Rule Ordering

You use **placement rules** to indicate whether applications are placed in specified pools, pools named by a user or group, or the default pool. To configure and order a set of rules:

1. Select the **Advanced** radio button on the Placement Rules tab.
2. Click **+** to add a new rule row and **-** to remove a rule row.
3. In each row, click  and select a rule. The available rules are:
 - specified pool; create the pool if it doesn't exist (default 1st)

- root.<username> pool; create the pool if it doesn't exist (default 2nd) - the application or query is placed into a pool with the name of the user who submitted it.
- specified pool only if the pool exists
- root.<username> pool only if the pool exists
- root.<primaryGroupName> pool; create the pool if it doesn't exist - the application or query is placed into a pool with the name of the primary group of the user who submitted it.
- root.<primaryGroupName> pool only if the pool exists
- root.<secondaryGroupName> pool only if one of these pools exists - the application or query is placed into a pool with a name that matches a secondary group of the user who submitted it.
- default pool; create the pool if it doesn't exist

For more information about these rules, see the description of the `queuePlacementPolicy` element in [Allocation File Format](#). Reorder rules by selecting different rules for existing rule rows. If a rule is always satisfied, subsequent rules are not evaluated and appear disabled.

4. Click **Save**. The [Fair Scheduler](#) allocation file (by default, `fair-scheduler.xml`) is updated.

Configuration Sets

A **configuration set** defines the allocation of resources across pools that may be active at a given time. For example, you can define "weekday" and "weekend" configuration sets, which define different resource pool configurations for different days of the week.

Creating a Configuration Set

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click the **Scheduling Rules** tab.
4. Click **Add Scheduling Rule**.
5. In the Configuration Set field, select the **Create New** radio button.
- 6.
7. Click **Add Configuration Set**. The Add Configuration Set dialog displays.
 - a. Type a name in the Name field and select the configuration set to clone from the **Clone from Configuration Set** drop-down.
 - b. Click **OK**. The configuration set is added to and selected in the **Configuration Sets** drop-down.
8. For each resource pool, click **Edit**.
 - a. Select a resource pool configuration set name.
 - b. Edit the pool properties and click **OK**.
9. Define one or more [scheduling rules](#) to specify when the configuration set is active.

Example Configuration Sets

The **weekday** configuration set assigns the **production** pool four times the resources of the **development** pool:

		+ Add Resource Pool		Default Settings		Configuration Sets		weekday	
Name	YARN						Impala		
	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Max Running Queries	Max Queued Queries	
root	1	100.0%	- / -	- / -	-	DRF	-	-	Edit
production	4	80.0%	- / -	- / -	-	DRF	-	-	Edit
development	1	20.0%	- / -	- / -	-	DRF	-	-	Edit

The **weekend** configuration set assigns the **production** and **development** pools an equal share of the resources:

		+ Add Resource Pool		Default Settings		Configuration Sets		weekend	
Name	YARN						Impala		
	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Max Running Queries	Max Queued Queries	
root	1	100.0%	- / -	- / -	-	DRF	-	-	Edit
production	1	50.0%	- / -	- / -	-	DRF	-	-	Edit
development	1	50.0%	- / -	- / -	-	DRF	-	-	Edit

The **default** configuration set assigns the **production** pool twice the resources of the **development** pool:

		+ Add Resource Pool		Default Settings		Configuration Sets		default	
Name	YARN						Impala		
	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Max Running Queries	Max Queued Queries	
root	1	100.0%	- / -	- / -	-	DRF	-	-	Edit
production	2	66.7%	- / -	- / -	-	DRF	-	-	Edit
development	1	33.3%	- / -	- / -	-	DRF	-	-	Edit

See [example scheduling rules](#) for these configuration sets.

Viewing the Properties of a Configuration Set

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. In the **Configuration Sets** drop-down, select a configuration set. The properties of each pool for that configuration set display.

Deleting a Configuration Set

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. In the **Configuration Sets** drop-down, select a configuration set. The properties of each pool for that configuration set display.
4. Click **Delete**.

Scheduling Rules

A **scheduling rule** defines when a configuration set is active. The configuration set is updated in affected services every hour.

Example Scheduling Rules

Consider the [example weekday and weekend](#) configuration sets. To specify that the **weekday** configuration set is active every weekday, **weekend** configuration set is active on the weekend (weekly on Saturday and Sunday), and the **default** configuration set is active all other times, define the following rules:

Scheduling Rule	Configuration Set
Repeats weekly on Monday, Tuesday, Wednesday, Thursday, Friday from 12:00 AM to 12:00 AM (PDT), starting 03/24/2014.	weekday Edit
Repeats weekly on Sunday, Saturday from 12:00 AM to 12:00 AM (PDT), starting 03/24/2014.	weekend Edit
Runs when all other rules don't apply.	default


Adding a Scheduling Rule

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click the **Scheduling Rules** tab.
4. Click **Add Scheduling Rule**.
5. In the **Configuration Set** drop-down, select a configuration set.
6. Choose whether the rule should repeat, the repeat frequency, and if the frequency is weekly, the repeat day or days.
7. If the schedule is not repeating, click the left side of the **on** field to display a drop-down calendar where you set the starting date and time. When you specify the date and time, a default time window of two hours is set in the right side of the **on** field. Click the right side to adjust the date and time.
8. Click **OK**.

Editing a Scheduling Rule

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click **Scheduling Rules**.
4. Click **Edit** at the right of a rule.
5. Edit the rule as desired.
6. Click **OK**.

Deleting a Scheduling Rule

1. Select **Clusters > Cluster name > Resource Management > Dynamic Resource Pools**. If the cluster has a YARN service, the Dynamic Resource Pools Status tab displays. If the cluster has only an Impala service enabled for dynamic resource pools, the Dynamic Resource Pools Configuration tab displays.
2. If the Status page is displayed, click the **Configuration** tab. A list of the currently configured pools with their configured limits displays.
3. Click **Scheduling Rules**.
4. Click  at the right of a rule and select **Delete**.

5. Click **OK**.

Managing Impala Admission Control

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

Admission control is an Impala feature that imposes limits on concurrent SQL queries, to avoid resource usage spikes and out-of-memory conditions on busy CDH clusters. It is a form of “throttling”. New queries are accepted and executed until certain conditions are met, such as too many queries or too much total memory used across the cluster. When one of these thresholds is reached, incoming queries wait to begin execution. These queries are queued and are admitted (that is, begin executing) when the resources become available.

For further information on Impala admission control, see [Admission Control and Query Queuing](#) on page 254.

Enabling and Disabling Impala Admission Control Using Cloudera Manager

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Category > Admission Control**.
4. Select or deselect the **Enable Impala Admission Control** checkbox.
5. Click **Save Changes** to commit the changes.
6. Restart the Impala service.

Configuring Impala Admission Control Using Cloudera Manager

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Category > Admission Control**.
4. Configure admission control properties:

`default_pool_max_queued`

Purpose: Maximum number of requests allowed to be queued before rejecting requests. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately or queue them, it is a soft limit; the overall number of queued queries might be slightly higher during times of heavy load. A negative value or 0 indicates requests are always rejected once the maximum concurrent requests are executing. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.

Type: int64

Default: 200

`default_pool_max_requests`

Purpose: Maximum number of concurrent outstanding requests allowed to run before incoming requests are queued. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately or queue them, it is a soft limit; the overall number of concurrent queries might be slightly higher during times of heavy load. A negative value indicates no limit. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.

Type: int64

Default: 200

`default_pool_mem_limit`

Purpose: Maximum amount of memory (across the entire cluster) that all outstanding requests in this pool can use before new requests to this pool are queued. Specified in bytes, megabytes, or gigabytes by a number followed by the suffix `b` (optional), `m`, or `g`, either uppercase or lowercase. You can specify floating-point values for megabytes and gigabytes, to represent fractional numbers such as `1.5`. You can also specify it as a percentage of the physical memory by specifying the suffix `%`. 0 or no setting indicates no limit. Defaults to bytes if no unit is given. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately

or queue them, it is a soft limit; the overall memory used by concurrent queries might be slightly higher during times of heavy load. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.



Note: Impala relies on the statistics produced by the `COMPUTE STATS` statement to estimate memory usage for each query. See [COMPUTE STATS Statement](#) for guidelines about how and when to use this statement.

Type: string

Default: "" (empty string, meaning unlimited)

`disable_admission_control`

Purpose: Turns off the admission control feature entirely, regardless of other configuration option settings.

Type: Boolean

Default: `true`

`disable_pool_max_requests`

Purpose: Disables all per-pool limits on the maximum number of running requests.

Type: Boolean

Default: `false`

`disable_pool_mem_limits`

Purpose: Disables all per-pool mem limits.

Type: Boolean

Default: `false`

`fair_scheduler_allocation_path`

Purpose: Path to the fair scheduler allocation file (`fair-scheduler.xml`).

Type: string

Default: "" (empty string)

Usage notes: Admission control only uses a small subset of the settings that can go in this file, as described below. For details about all the Fair Scheduler configuration settings, see the [Apache wiki](#).

`llama_site_path`

Purpose: Path to the Llama configuration file (`llama-site.xml`). If set, `fair_scheduler_allocation_path` must also be set.

Type: string

Default: "" (empty string)

Usage notes: Admission control only uses a small subset of the settings that can go in this file, as described below. For details about all the Llama configuration settings, see the [documentation on Github](#).

`queue_wait_timeout_ms`

Purpose: Maximum amount of time (in milliseconds) that a request waits to be admitted before timing out.

Type: `int64`

Default: `60000`

.

5. Click **Save Changes** to commit the changes.
6. Restart the Impala service.

Managing the Impala Llama ApplicationMaster



Note: Though Impala can be used together with YARN using simple configuration of static service pools in Cloudera Manager, the use of the general-purpose component Llama for integrated resource management within YARN is no longer supported with CDH 5.5 / Impala 2.3 and higher.

The Impala Llama ApplicationMaster (Llama) reserves and releases YARN-managed resources for Impala, thus reducing resource management overhead when performing Impala queries. Llama is used when you want to enable integrated resource management.

By default, YARN allocates resources bit-by-bit as needed by MapReduce jobs. Impala needs all resources available at the same time, so that intermediate results can be exchanged between cluster nodes, and queries do not stall partway through waiting for new resources to be allocated. Llama is the intermediary process that ensures all requested resources are available before each Impala query actually begins.

For more information about Llama, see [Llama - Low Latency Application Master](#).

For information on enabling Llama high availability, see [Llama High Availability](#) on page 343.

Enabling Integrated Resource Management Using Cloudera Manager

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

The Enable Integrated Resource Management wizard enables cgroups for the *all the hosts* in the cluster running Impala and YARN, adds one or more Llama roles to the Impala service, and configures the Impala and YARN services.

1. Start the wizard using one of the following paths:
 - Cluster-level
 1. Select **Clusters** > *Cluster name* > **Dynamic Resource Pools**.
 2. In the Status section, click **Enable**.
 - Service-level
 1. Go to the Impala service.
 2. Select **Actions** > **Enable Integrated Resource Management**.

The Enable Integrated Resource Management wizard starts and displays information about resource management options and the actions performed by the wizard.

2. Click **Continue**.
3. Leave the **Enable Cgroup-based Resource Management** checkbox checked and click **Continue**.
4. Click the **Impala Llama ApplicationMaster Hosts** field to display a dialog for choosing Llama hosts.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
 - Rack name
5. Specify or select one or more hosts and click **OK**.
 6. Click **Continue**. A progress screen displays with a summary of the wizard actions.

Resource Management

7. Click **Continue**.
8. Click **Restart Now** to restart the cluster and apply the configuration changes or click **leave this wizard** to restart at a later time.
9. Click **Finish**.

Disabling Integrated Resource Management Using Cloudera Manager

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

The Enable Integrated Resource Management wizard enables cgroups for the *all the hosts* in the cluster running Impala and YARN, adds one or more Llama roles to the Impala service, and configures the Impala and YARN services.

1. Start the wizard using one of the following paths:
 - Cluster-level
 1. Select **Clusters** > *Cluster name* > **Dynamic Resource Pools**.
 2. In the Status section, click **Disable**.
 - Service-level
 1. Go to the Impala service.
 2. Select **Actions** > **Disable Integrated Resource Management**.

The Disable Integrated Resource Management wizard starts and displays information about resource management options and the actions performed by the wizard.

Configuring Llama Using Cloudera Manager

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

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Impala Llama ApplicationMaster**.
4. Edit [configuration properties](#).
5. Click **Save Changes** to commit the changes.
6. Restart the Llama role.

Impala Resource Management

Impala supports two types of resource management: independent and integrated. Independent resource management is supported for CDH 4 and CDH 5 and is implemented by admission control. Integrated resource management is supported for CDH 5 and is implemented by YARN and Llama.



Note:

When using YARN with Impala, Cloudera recommends using the static partitioning technique (through a static service pool) rather than the combination of YARN and Llama. YARN is a central, synchronous scheduler and thus introduces higher latency and variance which is better suited for batch processing than for interactive workloads like Impala (especially with higher concurrency). Currently, YARN allocates memory throughout the query, making it hard to reason about out-of-memory and timeout conditions.

Admission Control and Query Queuing

Admission control is an Impala feature that imposes limits on concurrent SQL queries, to avoid resource usage spikes and out-of-memory conditions on busy CDH clusters. It is a form of “throttling”. New queries are accepted and executed until certain conditions are met, such as too many queries or too much total memory used across the cluster. When

one of these thresholds is reached, incoming queries wait to begin execution. These queries are queued and are admitted (that is, begin executing) when the resources become available.

In addition to the threshold values for currently executing queries, you can place limits on the maximum number of queries that are queued (waiting) and a limit on the amount of time they might wait before returning with an error. These queue settings let you ensure that queries do not wait indefinitely, so that you can detect and correct “starvation” scenarios.

Enable this feature if your cluster is underutilized at some times and overutilized at others. Overutilization is indicated by performance bottlenecks and queries being cancelled due to out-of-memory conditions, when those same queries are successful and perform well during times with less concurrent load. Admission control works as a safeguard to avoid out-of-memory conditions during heavy concurrent usage.



Important:

- Cloudera strongly recommends you upgrade to CDH 5 or higher to use admission control. In CDH 4, admission control will only work if you *do not* have Hue deployed; unclosed Hue queries will accumulate and exceed the queue size limit. On CDH 4, to use admission control, you must explicitly enable it by specifying `--disable_admission_control=false` in the `impalad` command-line options.
- Use the `COMPUTE STATS` statement for large tables involved in join queries, and follow other steps from [Tuning Impala for Performance](#) to tune your queries. Although `COMPUTE STATS` is an important statement to help optimize query performance, it is especially important when admission control is enabled:
 - When queries complete quickly and are tuned for optimal memory usage, there is less chance of performance or capacity problems during times of heavy load.
 - The admission control feature also relies on the statistics produced by the `COMPUTE STATS` statement to generate accurate estimates of memory usage for complex queries. If the estimates are inaccurate due to missing statistics, Impala might hold back queries unnecessarily even though there is sufficient memory to run them, or might allow queries to run that end up exceeding the memory limit and being cancelled.

Overview of Impala Admission Control

On a busy CDH cluster, you might find there is an optimal number of Impala queries that run concurrently. Because Impala queries are typically I/O-intensive, you might not find any throughput benefit in running more concurrent queries when the I/O capacity is fully utilized. Because Impala by default cancels queries that exceed the specified memory limit, running multiple large-scale queries at once can result in having to re-run some queries that are cancelled.

The admission control feature lets you set a cluster-wide upper limit on the number of concurrent Impala queries and on the memory used by those queries. Any additional queries are queued until the earlier ones finish, rather than being cancelled or running slowly and causing contention. As other queries finish, the queued queries are allowed to proceed.

For details on the internal workings of admission control, see [How Impala Schedules and Enforces Limits on Concurrent Queries](#) on page 256.

How Impala Admission Control Relates to YARN

The admission control feature is similar in some ways to the YARN resource management framework, and they can be used separately or together. This section describes some similarities and differences, to help you decide when to use one, the other, or both together.

Admission control is a lightweight, decentralized system that is suitable for workloads consisting primarily of Impala queries and other SQL statements. It sets “soft” limits that smooth out Impala memory usage during times of heavy load, rather than taking an all-or-nothing approach that cancels jobs that are too resource-intensive.

Because the admission control system is not aware of other Hadoop workloads such as MapReduce jobs, you might use YARN with static service pools on heterogeneous CDH 5 clusters where resources are shared between Impala and other Hadoop components. Devote a percentage of cluster resources to Impala, allocate another percentage for MapReduce and other batch-style workloads; let admission control handle the concurrency and memory usage for the Impala work within the cluster, and let YARN manage the remainder of work within the cluster.

You could also try out the combination of YARN, Impala, and Llama, where YARN manages all cluster resources and Impala queries request resources from YARN by using the Llama component as an intermediary. YARN is a more centralized, general-purpose service, with somewhat higher latency than admission control due to the requirement to pass requests back and forth through the YARN and Llama components.

The Impala admission control feature uses the same mechanism as the YARN resource manager to map users to pools and authenticate them. Although the YARN resource manager is only available with CDH 5 and higher, internally Impala includes the necessary infrastructure to work consistently on both CDH 4 and CDH 5. You do not need to run the YARN and Llama components for admission control to operate.

In Cloudera Manager, the controls for Impala resource management change slightly depending on whether the Llama role is enabled, which brings Impala under the control of YARN. When you use Impala without the Llama role, you can specify three properties (memory limit, query queue size, and queue timeout) for the admission control feature. When the Llama role is enabled, you can specify query queue size and queue timeout, but the memory limit is enforced by YARN and not settable through resource pools.

For full details about using Impala with YARN, see [Integrated Resource Management with YARN](#) on page 263.

How Impala Schedules and Enforces Limits on Concurrent Queries

The admission control system is decentralized, embedded in each Impala daemon and communicating through the statestore mechanism. Although the limits you set for memory usage and number of concurrent queries apply cluster-wide, each Impala daemon makes its own decisions about whether to allow each query to run immediately or to queue it for a less-busy time. These decisions are fast, meaning the admission control mechanism is low-overhead, but might be imprecise during times of heavy load. There could be times when the query queue contained more queries than the specified limit, or when the estimated of memory usage for a query is not exact and the overall memory usage exceeds the specified limit. Thus, you typically err on the high side for the size of the queue, because there is not a big penalty for having a large number of queued queries; and you typically err on the low side for the memory limit, to leave some headroom for queries to use more memory than expected, without being cancelled as a result.

At any time, the set of queued queries could include queries submitted through multiple different Impala daemon hosts. All the queries submitted through a particular host will be executed in order, so a `CREATE TABLE` followed by an `INSERT` on the same table would succeed. Queries submitted through different hosts are not guaranteed to be executed in the order they were received. Therefore, if you are using load-balancing or other round-robin scheduling where different statements are submitted through different hosts, set up all table structures ahead of time so that the statements controlled by the queuing system are primarily queries, where order is not significant. Or, if a sequence of statements needs to happen in strict order (such as an `INSERT` followed by a `SELECT`), submit all those statements through a single session, while connected to the same Impala daemon host.

The limit on the number of concurrent queries is a “soft” one. To achieve high throughput, Impala makes quick decisions at the host level about which queued queries to dispatch. Therefore, Impala might slightly exceed the limit from time to time.

To avoid a large backlog of queued requests, you can also set an upper limit on the size of the queue for queries that are delayed. When the number of queued queries exceeds this limit, further queries are cancelled rather than being queued. You can also configure a timeout period, after which queued queries are cancelled, to avoid indefinite waits. If a cluster reaches this state where queries are cancelled due to too many concurrent requests or long waits for query execution to begin, that is a signal for an administrator to take action, either by provisioning more resources, scheduling work on the cluster to smooth out the load, or by doing [Impala performance tuning](#) to enable higher throughput.

How Admission Control works with Impala Clients (JDBC, ODBC, HiveServer2)

Most aspects of admission control work transparently with client interfaces such as JDBC and ODBC:

- If a SQL statement is put into a queue rather than running immediately, the API call blocks until the statement is dequeued and begins execution. At that point, the client program can request to fetch results, which might also block until results become available.
- If a SQL statement is cancelled because it has been queued for too long or because it exceeded the memory limit during execution, the error is returned to the client program with a descriptive error message.

If you want to submit queries to different resource pools through the `REQUEST_POOL` query option, as described in [REQUEST_POOL Query Option](#), In Impala 2.0 and higher you can change that query option through a SQL `SET` statement that you submit from the client application, in the same session. Prior to Impala 2.0, that option was only settable for a session through the `impala-shell SET` command, or cluster-wide through an `impalad` startup option.

Admission control has the following limitations or special behavior when used with JDBC or ODBC applications:

- The `MEM_LIMIT` query option, sometimes useful to work around problems caused by inaccurate memory estimates for complicated queries, is only settable through the `impala-shell` interpreter and cannot be used directly through JDBC or ODBC applications.
- Admission control does not use the other resource-related query options, `RESERVATION_REQUEST_TIMEOUT` or `V_CPU_CORES`. Those query options only apply to the YARN resource management framework.

Configuring Admission Control

The configuration options for admission control range from the simple (a single resource pool with a single set of options) to the complex (multiple resource pools with different options, each pool handling queries for a different set of users and groups). You can configure the settings through the Cloudera Manager user interface, or on a system without Cloudera Manager by editing configuration files or through startup options to the `impalad` daemon.

Admission Control Options

The following Impala configuration options let you adjust the settings of the admission control feature. When supplying the options on the command line, prepend the option name with `--`.

`default_pool_max_queued`

Purpose: Maximum number of requests allowed to be queued before rejecting requests. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately or queue them, it is a soft limit; the overall number of queued queries might be slightly higher during times of heavy load. A negative value or 0 indicates requests are always rejected once the maximum concurrent requests are executing. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.

Type: `int64`

Default: 200

`default_pool_max_requests`

Purpose: Maximum number of concurrent outstanding requests allowed to run before incoming requests are queued. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately or queue them, it is a soft limit; the overall number of concurrent queries might be slightly higher during times of heavy load. A negative value indicates no limit. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.

Type: `int64`

Default: 200

`default_pool_mem_limit`

Purpose: Maximum amount of memory (across the entire cluster) that all outstanding requests in this pool can use before new requests to this pool are queued. Specified in bytes, megabytes, or gigabytes by a number followed by the suffix `b` (optional), `m`, or `g`, either uppercase or lowercase. You can specify floating-point values for megabytes and gigabytes, to represent fractional numbers such as `1.5`. You can also specify it as a percentage of the physical memory by specifying the suffix `%`. 0 or no setting indicates no limit. Defaults to bytes if no unit is given. Because this limit applies cluster-wide, but each Impala node makes independent decisions to run queries immediately or

queue them, it is a soft limit; the overall memory used by concurrent queries might be slightly higher during times of heavy load. Ignored if `fair_scheduler_config_path` and `llama_site_path` are set.



Note: Impala relies on the statistics produced by the `COMPUTE STATS` statement to estimate memory usage for each query. See [COMPUTE STATS Statement](#) for guidelines about how and when to use this statement.

Type: string

Default: "" (empty string, meaning unlimited)

`disable_admission_control`

Purpose: Turns off the admission control feature entirely, regardless of other configuration option settings.

Type: Boolean

Default: true

`disable_pool_max_requests`

Purpose: Disables all per-pool limits on the maximum number of running requests.

Type: Boolean

Default: false

`disable_pool_mem_limits`

Purpose: Disables all per-pool mem limits.

Type: Boolean

Default: false

`fair_scheduler_allocation_path`

Purpose: Path to the fair scheduler allocation file (`fair-scheduler.xml`).

Type: string

Default: "" (empty string)

Usage notes: Admission control only uses a small subset of the settings that can go in this file, as described below. For details about all the Fair Scheduler configuration settings, see the [Apache wiki](#).

`llama_site_path`

Purpose: Path to the Llama configuration file (`llama-site.xml`). If set, `fair_scheduler_allocation_path` must also be set.

Type: string

Default: "" (empty string)

Usage notes: Admission control only uses a small subset of the settings that can go in this file, as described below. For details about all the Llama configuration settings, see the [documentation on Github](#).

`queue_wait_timeout_ms`

Purpose: Maximum amount of time (in milliseconds) that a request waits to be admitted before timing out.

Type: int64

Default: 60000

Configuring Admission Control Using Cloudera Manager

In Cloudera Manager, you can configure pools to manage queued Impala queries, and the options for the limit on number of concurrent queries and how to handle queries that exceed the limit. For details, see [Managing Resources with Cloudera Manager](#).

See [Examples of Admission Control Configurations](#) on page 259 for a sample setup for admission control under Cloudera Manager.

Configuring Admission Control Using the Command Line

If you do not use Cloudera Manager, you use a combination of startup options for the Impala daemon, and optionally editing or manually constructing the configuration files `fair-scheduler.xml` and `llama-site.xml`.

For a straightforward configuration using a single resource pool named `default`, you can specify configuration options on the command line and skip the `fair-scheduler.xml` and `llama-site.xml` configuration files.

For an advanced configuration with multiple resource pools using different settings, set up the `fair-scheduler.xml` and `llama-site.xml` configuration files manually. Provide the paths to each one using the Impala daemon command-line options, `--fair_scheduler_allocation_path` and `--llama_site_path` respectively.

The Impala admission control feature only uses the Fair Scheduler configuration settings to determine how to map users and groups to different resource pools. For example, you might set up different resource pools with separate memory limits, and maximum number of concurrent and queued queries, for different categories of users within your organization. For details about all the Fair Scheduler configuration settings, see the [Apache wiki](#).

The Impala admission control feature only uses a small subset of possible settings from the `llama-site.xml` configuration file:

```
llama.am.throttling.maximum.placed.reservations.queue_name
llama.am.throttling.maximum.queued.reservations.queue_name
```

For details about all the Llama configuration settings, see [Llama Default Configuration](#).

See [Example Admission Control Configurations Using Configuration Files](#) on page 261 for sample configuration files for admission control using multiple resource pools, without Cloudera Manager.

Examples of Admission Control Configurations

Example Admission Control Configurations Using Cloudera Manager

For full instructions about configuring dynamic resource pools through Cloudera Manager, see [Dynamic Resource Pools](#) on page 242. The following examples demonstrate some important points related to the Impala admission control feature.

The following figure shows a sample of the Dynamic Resource Pools page in Cloudera Manager, accessed through the **Clusters > Cluster name > Resource Management > Dynamic Resource Pools** menu choice and then the **Configuration** tab. Numbers from all the resource pools are combined into the topmost `root` pool. The `default` pool is for users who are not assigned any other pool by the user-to-pool mapping settings. The `development` and `production` pools show how you can set different limits for different classes of users, for total memory, number of concurrent queries, and number of queries that can be queued.

Dynamic Resource Pools for Cluster 1

[Status](#)
[Configuration](#)

[Resource Pools](#)
[Scheduling Rules](#)
[Placement Rules](#)
[User Limits](#)
[Other Settings](#)

Applications can run in a pool based on the user, the group of the submitting user, as well as **specific** pools and the default pool.

Allocate resources across pools using weights, minimum, and maximum limits. Configuration sets allow switching on different weight and limit settings activated by user-defined schedules.

Pools can be nested, each level of which can support a different scheduler, such as FIFO or fair scheduler. Each pool can be configured to allow only a certain set of users and groups to access the pool.

[+ Add Resource Pool](#)
Configuration Sets
default ▾
[+ Add Configuration Set](#)

Name	YARN						Impala			
	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Scheduling Policy	Max Memory	Max Running Queries	Max Queued Queries	
root	1	100.0%	- / -	- / -	-	DRF	-	-	-	Edit ▾
default	1	33.3%	- / -	- / -	-	DRF	50000MB	10	50	Edit ▾
development	1	33.3%	- / -	- / -	-	DRF	200000MB	50	100	Edit ▾
production	1	33.3%	- / -	- / -	-	DRF	1000000MB	100	200	Edit ▾

Figure 4: Sample Settings for Cloudera Manager Dynamic Resource Pools Page

The following figure shows a sample of the Placement Rules page in Cloudera Manager, accessed through the **Clusters > Cluster name > Resource Management > Dynamic Resource Pools** menu choice and then the **Configuration > Placement Rules** tabs. The settings demonstrate a reasonable configuration of a pool named `default` to service all requests where the specified resource pool does not exist, is not explicitly set, or the user or group is not authorized for the specified pool.

Dynamic Resource Pools for Cluster 1

[Status](#)
[Configuration](#)

[Resource Pools](#)
[Scheduling Rules](#)
[Placement Rules](#)
[User Limits](#)
[Other Settings](#)

Applications can run in a pool based on the user, the group of the submitting user, as well as **specific** pools and the default pool. Configure how an application will determine in which pool it will run.

Basic
 Advanced

Specify the order in which rules are evaluated to determine in which pool an application will run.

If a rule is always satisfied, subsequent rules are not evaluated and appear disabled. If a rule has a condition that is not satisfied, subsequent rules are evaluated. When none of the rules apply, the application is rejected.

This rule is always satisfied.

Subsequent rules are not evaluated.

[Save](#)

Figure 5: Sample Settings for Cloudera Manager Placement Rules Page

Example Admission Control Configurations Using Configuration Files

For clusters not managed by Cloudera Manager, here are sample `fair-scheduler.xml` and `llama-site.xml` files that define resource pools equivalent to the ones in the preceding Cloudera Manager dialog. These sample files are stripped down: in a real deployment they might contain other settings for use with various aspects of the YARN and Llama components. The settings shown here are the significant ones for the Impala admission control feature.

`fair-scheduler.xml`:

Although Impala does not use the `vcores` value, you must still specify it to satisfy YARN requirements for the file contents.

Each `<aclSubmitApps>` tag (other than the one for `root`) contains a comma-separated list of users, then a space, then a comma-separated list of groups; these are the users and groups allowed to submit Impala statements to the corresponding resource pool.

If you leave the `<aclSubmitApps>` element empty for a pool, nobody can submit directly to that pool; child pools can specify their own `<aclSubmitApps>` values to authorize users and groups to submit to those pools.

```

<allocations>
  <queue name="root">
    <aclSubmitApps> </aclSubmitApps>
    <queue name="default">
      <maxResources>50000 mb, 0 vcores</maxResources>
      <aclSubmitApps>*</aclSubmitApps>
    </queue>
    <queue name="development">
      <maxResources>200000 mb, 0 vcores</maxResources>
      <aclSubmitApps>user1,user2 dev,ops,admin</aclSubmitApps>
    </queue>
    <queue name="production">
      <maxResources>1000000 mb, 0 vcores</maxResources>
      <aclSubmitApps> ops,admin</aclSubmitApps>
    </queue>
  </queue>

```

```

    <queuePlacementPolicy>
      <rule name="specified" create="false" />
      <rule name="default" />
    </queuePlacementPolicy>
  </allocations>

```

llama-site.xml:

```

<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>llama.am.throttling.maximum.placed.reservations.root.default</name>
    <value>10</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.queued.reservations.root.default</name>
    <value>50</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.placed.reservations.root.development</name>
    <value>50</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.queued.reservations.root.development</name>
    <value>100</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.placed.reservations.root.production</name>
    <value>100</value>
  </property>
  <property>
    <name>llama.am.throttling.maximum.queued.reservations.root.production</name>
    <value>200</value>
  </property>
</configuration>

```

Guidelines for Using Admission Control

To see how admission control works for particular queries, examine the profile output for the query. This information is available through the `PROFILE` statement in `impala-shell` immediately after running a query in the shell, on the **queries** page of the Impala debug web UI, or in the Impala log file (basic information at log level 1, more detailed information at log level 2). The profile output contains details about the admission decision, such as whether the query was queued or not and which resource pool it was assigned to. It also includes the estimated and actual memory usage for the query, so you can fine-tune the configuration for the memory limits of the resource pools.

Where practical, use Cloudera Manager to configure the admission control parameters. The Cloudera Manager GUI is much simpler than editing the configuration files directly.

Remember that the limits imposed by admission control are “soft” limits. Although the limits you specify for number of concurrent queries and amount of memory apply cluster-wide, the decentralized nature of this mechanism means that each Impala node makes its own decisions about whether to allow queries to run immediately or to queue them. These decisions rely on information passed back and forth between nodes by the statestore service. If a sudden surge in requests causes more queries than anticipated to run concurrently, then as a fallback, the overall Impala memory limit and the Linux cgroups mechanism serve as hard limits to prevent overallocation of memory, by cancelling queries if necessary.

If you have trouble getting a query to run because its estimated memory usage is too high, you can override the estimate by setting the `MEM_LIMIT` query option in `impala-shell`, then issuing the query through the shell in the same session. The `MEM_LIMIT` value is treated as the estimated amount of memory, overriding the estimate that Impala would generate based on table and column statistics. This value is used only for making admission control decisions, and is not pre-allocated by the query.

In `impala-shell`, you can also specify which resource pool to direct queries to by setting the `REQUEST_POOL` query option. (This option was named `YARN_POOL` during the CDH 5 beta period.)

The statements affected by the admission control feature are primarily queries, but also include statements that write data such as `INSERT` and `CREATE TABLE AS SELECT`. Most write operations in Impala are not resource-intensive, but inserting into a Parquet table can require substantial memory due to buffering 1 GB of data before writing out each Parquet data block. See [Loading Data into Parquet Tables](#) for instructions about inserting data efficiently into Parquet tables.

Although admission control does not scrutinize memory usage for other kinds of DDL statements, if a query is queued due to a limit on concurrent queries or memory usage, subsequent statements in the same session are also queued so that they are processed in the correct order:

```
-- This query could be queued to avoid out-of-memory at times of heavy load.
select * from huge_table join enormous_table using (id);
-- If so, this subsequent statement in the same session is also queued
-- until the previous statement completes.
drop table huge_table;
```

If you set up different resource pools for different users and groups, consider reusing any classifications and hierarchy you developed for use with Sentry security. See [Enabling Sentry Authorization for Impala](#) for details.

For details about all the Fair Scheduler configuration settings, see [Fair Scheduler Configuration](#), in particular the tags such as `<queue>` and `<aclSubmitApps>` to map users and groups to particular resource pools (queues).

Integrated Resource Management with YARN

You can limit the CPU and memory resources used by Impala, to manage and prioritize workloads on clusters that run jobs from many Hadoop components. (Currently, there is no limit or throttling on the I/O for Impala queries.) In CDH 5, Impala can use the underlying Apache Hadoop YARN resource management framework, which allocates the required resources for each Impala query. Impala estimates the resources required by the query on each host of the cluster, and requests the resources from YARN.

Requests from Impala to YARN go through an intermediary service called Llama. When the resource requests are granted, Impala starts the query and places all relevant execution threads into the cgroup containers and sets up the memory limit on each host. If sufficient resources are not available, the Impala query waits until other jobs complete and the resources are freed. During query processing, as the need for additional resources arises, Llama can “expand” already-requested resources, to avoid over-allocating at the start of the query.

After a query is finished, Llama caches the resources (for example, leaving memory allocated) in case they are needed for subsequent Impala queries. This caching mechanism avoids the latency involved in making a whole new set of resource requests for each query. If the resources are needed by YARN for other types of jobs, Llama returns them.

While the delays to wait for resources might make individual queries seem less responsive on a heavily loaded cluster, the resource management feature makes the overall performance of the cluster smoother and more predictable, without sudden spikes in utilization due to memory paging, CPUs pegged at 100%, and so on.



Note:

When using YARN with Impala, Cloudera recommends using the static partitioning technique (through a static service pool) rather than the combination of YARN and Llama. YARN is a central, synchronous scheduler and thus introduces higher latency and variance which is better suited for batch processing than for interactive workloads like Impala (especially with higher concurrency). Currently, YARN allocates memory throughout the query, making it hard to reason about out-of-memory and timeout conditions.

The Llama Daemon

Llama is a system that mediates resource management between Impala and Hadoop YARN. Llama enables Impala to reserve, use, and release resource allocations in a Hadoop cluster. Llama is only required if resource management is enabled in Impala.

By default, YARN allocates resources bit-by-bit as needed by MapReduce jobs. Impala needs all resources available at the same time, so that intermediate results can be exchanged between cluster nodes, and queries do not stall partway through waiting for new resources to be allocated. Llama is the intermediary process that ensures all requested resources are available before each Impala query actually begins.

For management through Cloudera Manager, see [The Impala Llama ApplicationMaster](#).

Controlling Resource Estimation Behavior

By default, Impala consults the table statistics and column statistics for each table in a query, and uses those figures to construct estimates of needed resources for each query. See [COMPUTE STATS Statement](#) for the statement to collect table and column statistics for a table.

To avoid problems with inaccurate or missing statistics, which can lead to inaccurate estimates of resource consumption, Impala allows you to set default estimates for CPU and memory resource consumption. As a query grows to require more resources, Impala will request more resources from Llama (this is called “expanding” a query reservation). When the query is complete, those resources are returned to YARN as normal. To enable this feature, use the command-line option `-rm_always_use_defaults` when starting `impalad`, and optionally `-rm_default_memory=size` and `-rm_default_cpu_cores`. Cloudera recommends always running with `-rm_always_use_defaults` enabled when using resource management, because if the query needs more resources than the default values, the resource requests are expanded dynamically as the query runs. See [impalad Startup Options for Resource Management](#) on page 265 for details about each option.

Checking Resource Estimates and Actual Usage

To make resource usage easier to verify, the output of the `EXPLAIN` SQL statement now includes information about estimated memory usage, whether table and column statistics are available for each table, and the number of virtual cores that a query will use. You can get this information through the `EXPLAIN` statement without actually running the query. The extra information requires setting the query option `EXPLAIN_LEVEL=verbose`; see [EXPLAIN Statement](#) for details. The same extended information is shown at the start of the output from the `PROFILE` statement in `impala-shell`. The detailed profile information is only available after running the query. You can take appropriate actions (gathering statistics, adjusting query options) if you find that queries fail or run with suboptimal performance when resource management is enabled.

How Resource Limits Are Enforced

- CPU limits are enforced by the Linux cgroups mechanism. YARN grants resources in the form of containers that correspond to cgroups on the respective machines.
- Memory is enforced by Impala's query memory limits. Once a reservation request has been granted, Impala sets the query memory limit according to the granted amount of memory before executing the query.

Enabling Resource Management for Impala

To enable resource management for Impala, first you [set up the YARN and Llama services for your CDH cluster](#). Then you [add startup options and customize resource management settings](#) for the Impala services.

Required CDH Setup for Resource Management with Impala

YARN is the general-purpose service that manages resources for many Hadoop components within a CDH cluster. Llama is a specialized service that acts as an intermediary between Impala and YARN, translating Impala resource requests to YARN and coordinating with Impala so that queries only begin executing when all needed resources have been granted by YARN.

For information about setting up the YARN and Llama services, see the instructions for [Cloudera Manager](#).

Using Impala with a Llama High Availability Configuration

Impala can take advantage of the Llama high availability (HA) feature, with additional Llama servers that step in if the primary one becomes unavailable. (Only one Llama server at a time services all resource requests.) Before using this feature from Impala, read the background information about Llama HA, its main features, and how to set it up.

Command-line method for systems without Cloudera Manager:

Setting up the Impala side in a Llama HA configuration involves setting the `impalad` configuration options

`-llama_addresses` (mandatory) and optionally `-llama_max_request_attempts`, `-llama_registration_timeout_secs`, and `-llama_registration_wait_secs`. See the next section [impalad Startup Options for Resource Management](#) on page 265 for usage instructions for those options.

The `impalad` daemon on the coordinator host registers with the Llama server for each query, receiving a handle that is used for subsequent resource requests. If a Llama server becomes unavailable, all running Impala queries are cancelled. Subsequent queries register with the next specified Llama server. This registration only happens when a query or similar request causes an `impalad` to request resources through Llama. Therefore, when a Llama server becomes unavailable, that fact might not be reported immediately in the Impala status information such as the `metrics` page in the debug web UI.

Cloudera Manager method: See [Llama High Availability](#).

impalad Startup Options for Resource Management

The following startup options for `impalad` enable resource management and customize its parameters for your cluster configuration:

- `-enable_rm`: Whether to enable resource management or not, either `true` or `false`. The default is `false`. None of the other resource management options have any effect unless `-enable_rm` is turned on.
- `-llama_host`: Hostname or IP address of the Llama service that Impala should connect to. The default is `127.0.0.1`.
- `-llama_port`: Port of the Llama service that Impala should connect to. The default is `15000`.
- `-llama_callback_port`: Port that Impala should start its Llama callback service on. Llama reports when resources are granted or preempted through that service.
- `-cgroup_hierarchy_path`: Path where YARN and Llama will create cgroups for granted resources. Impala assumes that the cgroup for an allocated container is created in the path `'cgroup_hierarchy_path + container_id'`.
- `-rm_always_use_defaults`: If this Boolean option is enabled, Impala ignores computed estimates and always obtains the default memory and CPU allocation from Llama at the start of the query. These default estimates are approximately 2 CPUs and 4 GB of memory, possibly varying slightly depending on cluster size, workload, and so on. Cloudera recommends enabling `-rm_always_use_defaults` whenever resource management is used, and relying on these default values (that is, leaving out the two following options).
- `-rm_default_memory=size`: Optionally sets the default estimate for memory usage for each query. You can use suffixes such as M and G for megabytes and gigabytes, the same as with the [MEM_LIMIT](#) query option. Only has an effect when `-rm_always_use_defaults` is also enabled.
- `-rm_default_cpu_cores`: Optionally sets the default estimate for number of virtual CPU cores for each query. Only has an effect when `-rm_always_use_defaults` is also enabled.

The following options fine-tune the interaction of Impala with Llama when Llama high availability (HA) is enabled. The `-llama_addresses` option is only applicable in a Llama HA environment. `-llama_max_request_attempts`, `-llama_registration_timeout_secs`, and `-llama_registration_wait_secs` work whether or not Llama HA is enabled, but are most useful in combination when Llama is set up for high availability.

- `-llama_addresses`: Comma-separated list of `hostname:port` items, specifying all the members of the Llama availability group. Defaults to `"127.0.0.1:15000"`.
- `-llama_max_request_attempts`: Maximum number of times a request to reserve, expand, or release resources is retried until the request is cancelled. Attempts are only counted after Impala is registered with Llama. That is, a request survives at most `llama_max_request_attempts-1` re-registrations. Defaults to 5.
- `-llama_registration_timeout_secs`: Maximum number of seconds that Impala will attempt to register or re-register with Llama. If registration is unsuccessful, Impala cancels the action with an error, which could result in an `impalad` startup failure or a cancelled query. A setting of `-1` means try indefinitely. Defaults to 30.
- `-llama_registration_wait_secs`: Number of seconds to wait between attempts during Llama registration. Defaults to 3.

impala-shell Query Options for Resource Management

Before issuing SQL statements through the `impala-shell` interpreter, you can use the `SET` command to configure the following parameters related to resource management:

Resource Management

- [EXPLAIN_LEVEL Query Option](#)
- [MEM_LIMIT Query Option](#)
- [RESERVATION_REQUEST_TIMEOUT Query Option \(CDH 5 only\)](#)
- [V_CPU_CORES Query Option \(CDH 5 only\)](#)

Limitations of Resource Management for Impala

Currently, Impala in CDH 5 has the following limitations for resource management of Impala queries:

- Table statistics are required, and column statistics are highly valuable, for Impala to produce accurate estimates of how much memory to request from YARN. See [Overview of Table Statistics](#) and [Overview of Column Statistics](#) for instructions on gathering both kinds of statistics, and [EXPLAIN Statement](#) for the extended `EXPLAIN` output where you can check that statistics are available for a specific table and set of columns.
- If the Impala estimate of required memory is lower than is actually required for a query, Impala dynamically expands the amount of requested memory. Queries might still be cancelled if the reservation expansion fails, for example if there are insufficient remaining resources for that pool, or the expansion request takes long enough that it exceeds the query timeout interval, or because of YARN preemption. You can see the actual memory usage after a failed query by issuing a `PROFILE` command in `impala-shell`. Specify a larger memory figure with the `MEM_LIMIT` query option and re-try the query.

The `MEM_LIMIT` query option, and the other resource-related query options, are settable through the ODBC or JDBC interfaces in Impala 2.0 and higher. This is a former limitation that is now lifted.

Performance Management

This section describes mechanisms and best practices for improving performance.

Related Information

- [Tuning Impala for Performance](#)

Optimizing Performance in CDH

This section provides solutions to some performance problems, and describes configuration best practices.



Important: If you are running CDH over 10 Gbps Ethernet, improperly set network configuration or improperly applied NIC firmware or drivers can noticeably degrade performance. Work with your network engineers and hardware vendors to make sure that you have the proper NIC firmware, drivers, and configurations in place and that your network performs properly. Cloudera recognizes that network setup and upgrade are challenging problems, and will make best efforts to share any helpful experiences.

Disabling Transparent Hugepage Compaction

Most Linux platforms supported by CDH 5 include a feature called **transparent hugepage compaction** which interacts poorly with Hadoop workloads and can seriously degrade performance.

Symptom: `top` and other system monitoring tools show a large percentage of the CPU usage classified as "system CPU". If system CPU usage is 30% or more of the total CPU usage, your system may be experiencing this issue.

What to do:



Note: In the following instructions, `defrag_file_pathname` depends on your operating system:

- Red Hat/CentOS: `/sys/kernel/mm/redhat_transparent_hugepage/defrag`
- Ubuntu/Debian, OL, SLES: `/sys/kernel/mm/transparent_hugepage/defrag`

1. To see whether transparent hugepage compaction is enabled, run the following command and check the output:

```
$ cat defrag_file_pathname
```

- `[always] never` means that transparent hugepage compaction is enabled.
- `always [never]` means that transparent hugepage compaction is disabled.

2. To disable transparent hugepage compaction, add the following command to `/etc/rc.local`:

```
echo never > defrag_file_pathname
```

You can also disable transparent hugepage compaction interactively (but remember this will not survive a reboot).

To disable transparent hugepage compaction temporarily as root:

```
# echo 'never' > defrag_file_pathname
```

To disable transparent hugepage compaction temporarily using sudo:

```
$ sudo sh -c "echo 'never' > defrag_file_pathname"
```

Setting the `vm.swappiness` Linux Kernel Parameter

`vm.swappiness` is a Linux kernel parameter that controls how aggressively memory pages are swapped to disk. It can be set to a value between 0-100; the higher the value, the more aggressive the kernel is in seeking out inactive memory pages and swapping them to disk.

You can see what value `vm.swappiness` is currently set to by looking at `/proc/sys/vm`; for example:

```
cat /proc/sys/vm/swappiness
```

On most systems, it is set to 60 by default. This is not suitable for Hadoop cluster nodes, because it can cause processes to get swapped out even when there is free memory available. This can affect stability and performance, and may cause problems such as lengthy garbage collection pauses for important system daemons. Cloudera recommends that you set this parameter to 10 or less; for example:

```
# sysctl -w vm.swappiness=10
```

Cloudera previously recommended a setting of 0, but in recent kernels (such as those included with RedHat 6.4 and higher, and Ubuntu 12.04 LTS and higher) a setting of 0 might lead to out of memory issues per this blog post: <http://www.percona.com/blog/2014/04/28/oom-relation-vm-swappiness0-new-kernel/>.

Improving Performance in Shuffle Handler and IFile Reader

The MapReduce shuffle handler and IFile reader use native Linux calls (`posix_fadvise(2)` and `sync_data_range`) on Linux systems with Hadoop native libraries installed. The subsections that follow provide details.

Shuffle Handler

You can improve MapReduce shuffle handler performance by enabling shuffle readahead. This causes the TaskTracker or Node Manager to pre-fetch map output before sending it over the socket to the reducer.

- To enable this feature for YARN, set the `mapreduce.shuffle.manage.os.cache` property to `true` (default). To further tune performance, adjust the value of the `mapreduce.shuffle.readahead.bytes` property. The default value is 4MB.
- To enable this feature for MRv1, set the `mapred.tasktracker.shuffle.fadvise` property to `true` (default). To further tune performance, adjust the value of the `mapred.tasktracker.shuffle.readahead.bytes` property. The default value is 4MB.

IFile Reader

Enabling IFile readahead increases the performance of merge operations. To enable this feature for either MRv1 or YARN, set the `mapreduce.ifile.readahead` property to `true` (default). To further tune the performance, adjust the value of the `mapreduce.ifile.readahead.bytes` property. The default value is 4MB.

Best Practices for MapReduce Configuration

The configuration settings described below can reduce inherent latencies in MapReduce execution. You set these values in `mapred-site.xml`.

Send a heartbeat as soon as a task finishes

Set the `mapreduce.tasktracker.outofband.heartbeat` property to `true` to let the TaskTracker send an out-of-band heartbeat on task completion to reduce latency; the default value is `false`:

```
<property>
  <name>mapreduce.tasktracker.outofband.heartbeat</name>
  <value>true</value>
</property>
```

Reduce the interval for JobClient status reports on single node systems

The `jobclient.progress.monitor.poll.interval` property defines the interval (in milliseconds) at which JobClient reports status to the console and checks for job completion. The default value is 1000 milliseconds; you may want to set this to a lower value to make tests run faster on a single-node cluster. Adjusting this value on a large production cluster may lead to unwanted client-server traffic.

```
<property>
  <name>jobclient.progress.monitor.poll.interval</name>
  <value>10</value>
</property>
```

Tune the JobTracker heartbeat interval

Tuning the minimum interval for the TaskTracker-to-JobTracker heartbeat to a smaller value may improve MapReduce performance on small clusters.

```
<property>
  <name>mapreduce.jobtracker.heartbeat.interval.min</name>
  <value>10</value>
</property>
```

Start MapReduce JVMs immediately

The `mapred.reduce.slowstart.completed.maps` property specifies the proportion of Map tasks in a job that must be completed before any Reduce tasks are scheduled. For small jobs that require fast turnaround, setting this value to 0 can improve performance; larger values (as high as 50%) may be appropriate for larger jobs.

```
<property>
  <name>mapred.reduce.slowstart.completed.maps</name>
  <value>0</value>
</property>
```

Tips and Best Practices for Jobs

This section describes changes you can make at the job level.

Use the Distributed Cache to Transfer the Job JAR

Use the distributed cache to transfer the job JAR rather than using the `JobConf(Class)` constructor and the `JobConf.setJar()` and `JobConf.setJarByClass()` methods.

To add JARs to the classpath, use `-libjars jar1, jar2`, which will copy the local JAR files to HDFS and then use the distributed cache mechanism to make sure they are available on the task nodes and are added to the task classpath.

The advantage of this over `JobConf.setJar` is that if the JAR is on a task node it won't need to be copied again if a second task from the same job runs on that node, though it will still need to be copied from the launch machine to HDFS.



Note: `-libjars` works only if your MapReduce driver uses [ToolRunner](#). If it doesn't, you would need to use the DistributedCache APIs (Cloudera does not recommend this).

For more information, see item 1 in the blog post [How to Include Third-Party Libraries in Your MapReduce Job](#).

Changing the Logging Level on a Job (MRv1)

You can change the logging level for an individual job. You do this by setting the following properties in the job configuration (`JobConf`):

- `mapreduce.map.log.level`
- `mapreduce.reduce.log.level`

Valid values are NONE, INFO, WARN, DEBUG, TRACE, and ALL.

Example:

```

JobConf conf = new JobConf();
...
conf.set("mapreduce.map.log.level", "DEBUG");
conf.set("mapreduce.reduce.log.level", "TRACE");
...

```

Choosing a Data Compression Format

Whether to compress your data and which compression formats to use can have a significant impact on performance. Two of the most important places to consider data compression are in terms of MapReduce jobs and data stored in HBase. For the most part, the principles are similar for each.

General Guidelines

- You need to balance the processing capacity required to compress and uncompress the data, the disk IO required to read and write the data, and the network bandwidth required to send the data across the network. The correct balance of these factors depends upon the characteristics of your cluster and your data, as well as your usage patterns.
- Compression is not recommended if your data is already compressed (such as images in JPEG format). In fact, the resulting file can actually be larger than the original.
- GZIP compression uses more CPU resources than Snappy or LZO, but provides a higher compression ratio. GZip is often a good choice for *cold data*, which is accessed infrequently. Snappy or LZO are a better choice for *hot data*, which is accessed frequently.
- BZip2 can also produce more compression than GZip for some types of files, at the cost of some speed when compressing and decompressing. HBase does not support BZip2 compression.
- Snappy often performs better than LZO. It is worth running tests to see if you detect a significant difference.
- For MapReduce, if you need your compressed data to be splittable, BZip2 and LZO formats can be split. Snappy and GZip blocks are not splittable, but files with Snappy blocks inside a container file format such as SequenceFile or Avro can be split. Snappy is intended to be used with a container format, like SequenceFiles or Avro data files, rather than being used directly on plain text, for example, since the latter is not splittable and cannot be processed in parallel using MapReduce. Splittability is not relevant to HBase data.
- For MapReduce, you can compress either the intermediate data, the output, or both. Adjust the parameters you provide for the MapReduce job accordingly. The following examples compress both the intermediate data and the output. MR2 is shown first, followed by MR1.

– MR2

```

hadoop jar hadoop-examples-.jar sort "-Dmapreduce.compress.map.output=true"
"-Dmapreduce.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
"-Dmapreduce.output.compress=true"
"-Dmapreduce.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
-outKey
org.apache.hadoop.io.Text -outValue org.apache.hadoop.io.Text input output

```

– MR1

```

hadoop jar hadoop-examples-.jar sort "-Dmapred.compress.map.output=true"
"-Dmapred.map.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec"
"-Dmapred.output.compress=true"
"-Dmapred.output.compression.codec=org.apache.hadoop.io.compress.GzipCodec" -outKey
org.apache.hadoop.io.Text -outValue org.apache.hadoop.io.Text input output

```

Configuring Data Compression Using Cloudera Manager

To configure support for LZO using Cloudera Manager, you must install the GPL Extras package, then configure services to use it. See [Installing GPL Extras](#) and [Configuring Services to Use the GPL Extras Parcel](#) on page 234.

Configuring Data Compression Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

To configure support for LZO in CDH, see [Step 5: \(Optional\) Install LZO](#) and [Configuring LZO](#). Snappy support is included in CDH.

To use Snappy in a MapReduce job, see [Using Snappy for MapReduce Compression](#). Use the same method for LZO, with the `codec.com.hadoop.compression.lzo.LzopCodec` instead.

Further Reading

For more information about compression algorithms in Hadoop, see section 4.2 of *Hadoop: The Definitive Guide*, by Tom White.

Tuning the Solr Server

Solr performance tuning is a complex task. The following sections provide more details.

Tuning to Complete During Setup

Some tuning is best completed during the setup of your system or may require some re-indexing.

Configuring Lucene Version Requirements

You can configure Solr to use a specific version of Lucene. This can help ensure that the Lucene version that Search uses includes the latest features and bug fixes. At the time that a version of Solr ships, Solr is typically configured to use the appropriate Lucene version, in which case there is no need to change this setting. If a subsequent Lucene update occurs, you can configure the Lucene version requirements by directly editing the `luceneMatchVersion` element in the `solrconfig.xml` file. Versions are typically of the form `x.y`, such as `4.4`. For example, to specify version `4.4`, you would ensure the following setting exists in `solrconfig.xml`:

```
<luceneMatchVersion>4.4</luceneMatchVersion>
```

Designing the Schema

When constructing a schema, use data types that most accurately describe the data that the fields will contain. For example:

- Use the `tdate` type for dates. Do this instead of representing dates as strings.
- Consider using the `text` type that applies to your language, instead of using `String`. For example, you might use `text_en`. Text types support returning results for subsets of an entry. For example, querying on "john" would find "John Smith", whereas with the string type, only exact matches are returned.
- For IDs, use the string type.

General Tuning

The following tuning categories can be completed at any time. It is less important to implement these changes before beginning to use your system.

General Tips

- Enabling multi-threaded faceting can provide better performance for field faceting. When multi-threaded faceting is enabled, field faceting tasks are completed in a parallel with a thread working on every field faceting task simultaneously. Performance improvements do not occur in all cases, but improvements are likely when all of the following are true:
 - The system uses highly concurrent hardware.
 - Faceting operations apply to large data sets over multiple fields.
 - There is not an unusually high number of queries occurring simultaneously on the system. Systems that are lightly loaded or that are mainly engaged with ingestion and indexing may be helped by multi-threaded faceting; for example, a system ingesting articles and being queried by a researcher. Systems heavily loaded by user queries are less likely to be helped by multi-threaded faceting; for example, an e-commerce site with heavy user-traffic.



Note: Multi-threaded faceting only applies to field faceting and not to query faceting.

- Field faceting identifies the number of unique entries for a field. For example, multi-threaded faceting could be used to simultaneously facet for the number of unique entries for the fields, "color" and "size". In such a case, there would be two threads, and each thread would work on faceting one of the two fields.
- Query faceting identifies the number of unique entries that match a query for a field. For example, query faceting could be used to find the number of unique entries in the "size" field are between 1 and 5. Multi-threaded faceting does not apply to these operations.

To enable multi-threaded faceting, add `facet-threads` to queries. For example, to use up to 1000 threads, you might use a query as follows:

```
http://localhost:8983/solr/collection1/select?q=*:*&facet=true&fl=id&facet.field=f0_ws&facet.threads=1000
```

If `facet-threads` is omitted or set to 0, faceting is single-threaded. If `facet-threads` is set to a negative value, such as -1, multi-threaded faceting will use as many threads as there are fields to facet up to the maximum number of threads possible on the system.

- If your environment does not require Near Real Time (NRT), turn off soft auto-commit in `solrconfig.xml`.
- In most cases, do not change the default batch size setting of 1000. If you are working with especially large documents, you may consider decreasing the batch size.
- To help identify any garbage collector (GC) issues, enable GC logging in production. The overhead is low and the JVM supports GC log rolling as of 1.6.0_34.
 - The minimum recommended GC logging flags are: `-XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+PrintGCDetails`.
 - To rotate the GC logs: `-Xloggc: -XX:+UseGCLogFileRotation -XX:NumberOfGCLogFiles=-XX:GCLogFileSize=`.

Solr and HDFS - the Block Cache



Warning: Do not enable the Solr HDFS write cache, because it can lead to [index corruption](#).

Cloudera Search enables Solr to store indexes in an HDFS filesystem. To maintain performance, an HDFS block cache has been implemented using Least Recently Used (LRU) semantics. This enables Solr to cache HDFS index files on read and write, storing the portions of the file in JVM direct memory (off heap) by default, or optionally in the JVM heap.

Batch jobs typically do not use the cache, while Solr servers (when serving queries or indexing documents) should. When running indexing using MapReduce, the MR jobs themselves do not use the block cache. Block write caching is turned off by default and should be left disabled.

Tuning of this cache is complex and best practices are continually being refined. In general, allocate a cache that is about 10-20% of the amount of memory available on the system. For example, when running HDFS and Solr on a host with 50 GB of memory, typically allocate 5-10 GB of memory using `solr.hdfs.blockcache.slab.count`. As index sizes grow you may need to tune this parameter to maintain optimal performance.



Note: Block cache metrics are currently unavailable.

Configuration

The following parameters control caching. They can be configured at the Solr process level by setting the respective system property or by editing the `solrconfig.xml` directly.

Parameter	Default	Description
<code>solr.hdfs.blockcache.enabled</code>	true	Enable the block cache.
<code>solr.hdfs.blockcache.read.enabled</code>	true	Enable the read cache.
<code>solr.hdfs.blockcache.write.enabled</code>	false	Enable the write cache. <div data-bbox="1036 982 1425 1150" style="border: 1px solid #f08080; padding: 5px; margin-top: 10px;"> <p>Warning: Do not enable the Solr HDFS write cache, because it can lead to index corruption.</p> </div>
<code>solr.hdfs.blockcache.direct.memory.allocation</code>	true	Enable direct memory allocation. If this is false, heap is used.
<code>solr.hdfs.blockcache.slab.count</code>	1	Number of memory slabs to allocate. Each slab is 128 MB in size.
<code>solr.hdfs.blockcache.global</code>	true	If enabled, one HDFS block cache is used for each collection on a host. If <code>blockcache.global</code> is disabled, each SolrCore on a host creates its own private HDFS block cache. Enabling this parameter simplifies managing HDFS block cache memory.

**Note:**

Increasing the direct memory cache size may make it necessary to increase the maximum direct memory size allowed by the JVM. Add the following to `/etc/default/solr` or `/opt/cloudera/parcels/CDH-*/etc/default/solr` to do so. You must also replace `MAXMEM` with a reasonable upper limit. A typical default JVM value for this is 64 MB. When using `MAXMEM`, you must specify a unit such as `g` for gigabytes or `m` for megabytes. If `MAXMEM` were set to 2, the following command would set `MaxDirectMemorySize` to 2 GB:

```
CATALINA_OPTS="-XX:MaxDirectMemorySize=MAXMEMg -XX:+UseLargePages"
```

Each Solr slab allocates the slab's memory, which is 128 MB by default, as well as allocating some additional direct memory overhead. Therefore, ensure that the `MaxDirectMemorySize` is set comfortably above the value expected for slabs alone. The amount of additional memory required varies according to multiple factors, but for most cases, setting `MaxDirectMemorySize` to at least 20-30% more than the total memory configured for slabs is sufficient. Setting the `MaxDirectMemorySize` to the number of slabs multiplied by the slab size does not provide enough memory.

Restart Solr servers after editing parameters.

Solr HDFS optimizes caching when performing NRT indexing using Lucene's `NRTCachingDirectory`.

Lucene caches a newly created segment if both of the following conditions are true:

- The segment is the result of a flush or a merge and the estimated size of the merged segment is \leq `solr.hdfs.nrtcachingdirectory.maxmergesizemb`.
- The total cached bytes is \leq `solr.hdfs.nrtcachingdirectory.maxcachedmb`.

The following parameters control NRT caching behavior:

Parameter	Default	Description
<code>solr.hdfs.nrtcachingdirectory.enable</code>	true	Whether to enable the <code>NRTCachingDirectory</code> .
<code>solr.hdfs.nrtcachingdirectory.maxcachedmb</code>	192	Size of the cache in megabytes.
<code>solr.hdfs.nrtcachingdirectory.maxmergesizemb</code>	16	Maximum segment size to cache.

Here is an example of `solrconfig.xml` with defaults:

```
<directoryFactory name="DirectoryFactory">
  <bool name="solr.hdfs.blockcache.enabled">${solr.hdfs.blockcache.enabled:true}</bool>
  <int name="solr.hdfs.blockcache.slabs.count">${solr.hdfs.blockcache.slabs.count:1}</int>
  <bool
name="solr.hdfs.blockcache.direct.memory.allocation">${solr.hdfs.blockcache.direct.memory.allocation:true}</bool>
  <int
name="solr.hdfs.blockcache.blocksperbank">${solr.hdfs.blockcache.blocksperbank:16384}</int>
  <bool
name="solr.hdfs.blockcache.read.enabled">${solr.hdfs.blockcache.read.enabled:true}</bool>
  <bool
name="solr.hdfs.nrtcachingdirectory.enable">${solr.hdfs.nrtcachingdirectory.enable:true}</bool>
  <int
name="solr.hdfs.nrtcachingdirectory.maxmergesizemb">${solr.hdfs.nrtcachingdirectory.maxmergesizemb:16}</int>
  <int
```

```
name="solr.hdfs.nrtcachingdirectory.maxcachedmb">${solr.hdfs.nrtcachingdirectory.maxcachedmb:192}</int>
</directoryFactory>
```

The following example illustrates passing Java options by editing the `/etc/default/solr` or `/opt/cloudera/parcels/CDH-*/etc/default/solr` configuration file:

```
CATALINA_OPTS="-Xmx10g -XX:MaxDirectMemorySize=20g -XX:+UseLargePages
-Dsolr.hdfs.blockcache.slab.count=100"
```

For better performance, Cloudera recommends setting the Linux swap space on all Solr server hosts as shown below:

```
# minimize swappiness
sudo sysctl vm.swappiness=1
sudo bash -c 'echo "vm.swappiness=1">> /etc/sysctl.conf'
# disable swap space until next reboot:
sudo /sbin/swapoff -a
```

Cloudera previously recommended a setting of 0, but in recent kernels (such as those included with RedHat 6.4 and higher, and Ubuntu 12.04 LTS and higher) a setting of 0 might lead to out of memory issues per this blog post: <http://www.percona.com/blog/2014/04/28/oom-relation-vm-swappiness0-new-kernel/>.

Threads

Configure the Tomcat server to have more threads per Solr instance. Note that this is only effective if your hardware is sufficiently powerful to accommodate the increased threads. 10,000 threads is a reasonable number to try in many cases.

To change the maximum number of threads, add a `maxThreads` element to the Connector definition in the Tomcat server's `server.xml` configuration file. For example, if you installed Search for CDH 5 using parcels installation, you would modify the Connector definition in the `<parcel path>/CDH/etc/solr/tomcat-conf.dist/conf/server.xml` file so this:

```
<Connector port="${solr.port}" protocol="HTTP/1.1"
  connectionTimeout="20000"
  redirectPort="8443" />
```

Becomes this:

```
<Connector port="${solr.port}" protocol="HTTP/1.1"
  maxThreads="10000"
  connectionTimeout="20000"
  redirectPort="8443" />
```

Garbage Collection

Choose different garbage collection options for best performance in different environments. Some garbage collection options typically chosen include:

- **Concurrent low pause collector:** Use this collector in most cases. This collector attempts to minimize "Stop the World" events. Avoiding these events can reduce connection timeouts, such as with ZooKeeper, and may improve user experience. This collector is enabled using `-XX:+UseConcMarkSweepGC`.
- **Throughput collector:** Consider this collector if raw throughput is more important than user experience. This collector typically uses more "Stop the World" events so this may negatively affect user experience and connection timeouts such as ZooKeeper heartbeats. This collector is enabled using `-XX:+UseParallelGC`. If `UseParallelGC` "Stop the World" events create problems, such as ZooKeeper timeouts, consider using the `UseParNewGC` collector as an alternative collector with similar throughput benefits.

You can also affect garbage collection behavior by increasing the Eden space to accommodate new objects. With additional Eden space, garbage collection does not need to run as frequently on new objects.

Replicated Data

You can adjust the degree to which different data is replicated.

Replicas

If you have sufficient additional hardware, add more replicas for a linear boost of query throughput. Note that adding replicas may slow write performance on the first replica, but otherwise this should have minimal negative consequences.

Transaction Log Replication

Beginning with CDH 5.4.1, Search for CDH supports configurable transaction log replication levels for replication logs stored in HDFS.

Configure the replication factor by modifying the `tlogDfsReplication` setting in `solrconfig.xml`. The `tlogDfsReplication` is a new setting in the `updateLog` settings area. An excerpt of the `solrconfig.xml` file where the transaction log replication factor is set is as follows:

```
<updateHandler class="solr.DirectUpdateHandler2">

  <!-- Enables a transaction log, used for real-time get, durability, and
        and solr cloud replica recovery. The log can grow as big as
        uncommitted changes to the index, so use of a hard autoCommit
        is recommended (see below).
        "dir" - the target directory for transaction logs, defaults to the
               solr data directory. -->
  <updateLog>
    <str name="dir">${solr.ulog.dir}</str>
    <int name="tlogDfsReplication">3</int>
  </updateLog>
```

You might want to increase the replication level from the default level of 1 to some higher value such as 3. Increasing the transaction log replication level can:

- Reduce the chance of data loss, especially when the system is otherwise configured to have single replicas of shards. For example, having single replicas of shards is reasonable when `autoAddReplicas` is enabled, but without additional transaction log replicas, the risk of data loss during a node failure would increase.
- Facilitate rolling upgrade of HDFS while Search is running. If you have multiple copies of the log, when a node with the transaction log becomes unavailable during the rolling upgrade process, another copy of the log can continue to collect transactions.
- Facilitate HDFS write lease recovery.

Initial testing shows no significant performance regression for common use cases.

Shards

In some cases, oversharding can help improve performance including intake speed. If your environment includes massively parallel hardware and you want to use these available resources, consider oversharding. You might increase the number of replicas per host from 1 to 2 or 3. Making such changes creates complex interactions, so you should continue to monitor your system's performance to ensure that the benefits of oversharding do not outweigh the costs.

Commits

Changing commit values may improve performance in some situation. These changes result in tradeoffs and may not be beneficial in all cases.

- For hard commit values, the default value of 60000 (60 seconds) is typically effective, though changing this value to 120 seconds may improve performance in some cases. Note that setting this value to higher values, such as 600 seconds may result in undesirable performance tradeoffs.
- Consider increasing the auto-commit value from 15000 (15 seconds) to 120000 (120 seconds).
- Enable soft commits and set the value to the largest value that meets your requirements. The default value of 1000 (1 second) is too aggressive for some environments.

Other Resources

- General information on Solr caching is available on the [SolrCaching](#) page on the Solr Wiki.
- Information on issues that influence performance is available on the [SolrPerformanceFactors](#) page on the Solr Wiki.
- [Resource Management](#) describes how to use Cloudera Manager to manage resources, for example with Linux cgroups.
- For information on improving querying performance, see [How to make searching faster](#).
- For information on improving indexing performance, see [How to make indexing faster](#).

Tuning Spark Applications

Optimizing Spark Performance

CDH 5.3 introduces a performance optimization that causes Spark to prefer RDDs that are already cached locally in HDFS. Using locally cached RDDs is important enough that Spark will wait a short time for the executors near these caches to be free. Spark does not start executors on hosts with cached data, and there is no further chance to select them during the task-matching phase. This is not a problem for most workloads, since most workloads start executors on most or all hosts in the cluster. However, if you do have problems with the optimization, the following constructor:

```
@DeveloperApi
def this(config: SparkConf, preferredNodeLocationData: Map[ String, Set[SplitInfo]])
= {
  this(config)
  this.preferredNodeLocationData = preferredNodeLocationData
}
```

allows you to explicitly specify the preferred locations to start executors. The following example from [examples/src/main/scala/org/apache/spark/examples/SparkHdfsLR.scala](#), provides an example of using this API:

```
...
val sparkConf = new SparkConf().setAppName("SparkHdfsLR")
val inputPath = args(0)
val conf = SparkHadoopUtil.get.newConfiguration()
val sc = new SparkContext(sparkConf,
  InputFormatInfo.computePreferredLocations(
    Seq(new InputFormatInfo(conf, classOf[org.apache.hadoop.mapred.TextInputFormat],
inputPath)))
...

```

Tuning YARN

This topic applies to YARN clusters only, and describes how to tune and optimize YARN for your cluster.

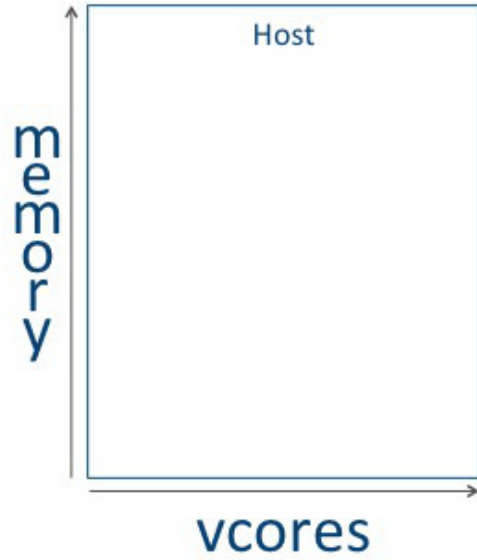


Note: Download the Cloudera [YARN tuning spreadsheet](#) to help calculate YARN configurations. For a short video overview, see [Tuning YARN Applications](#).

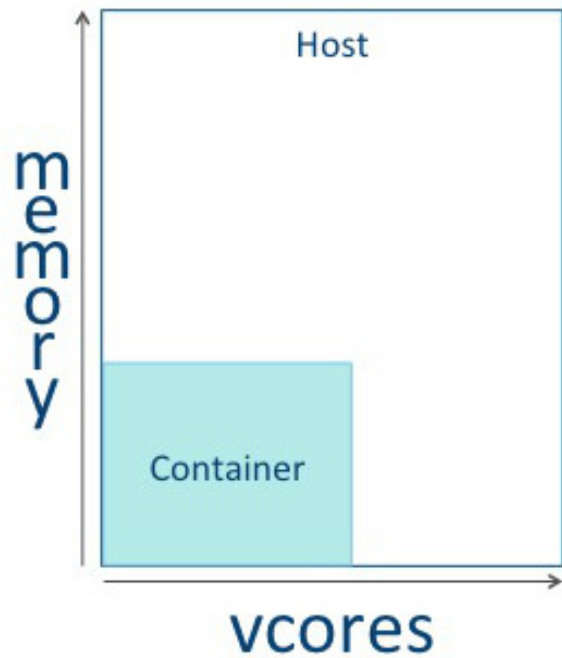
Overview

This overview provides an abstract description of a YARN cluster and the goals of YARN tuning.

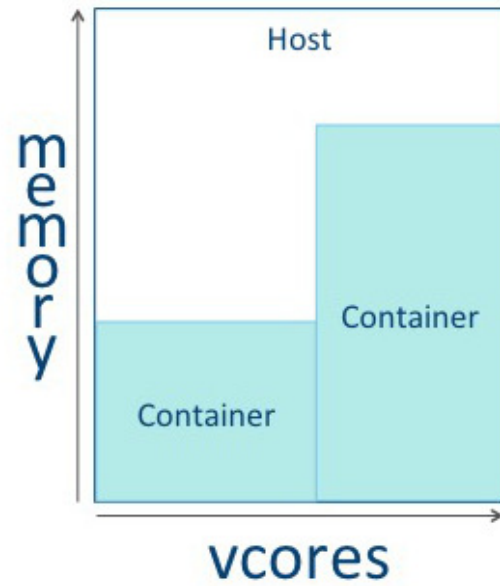
A YARN cluster is composed of host machines. Hosts provide memory and CPU resources. A *vc*ore, or virtual core, is a usage share of a host CPU.



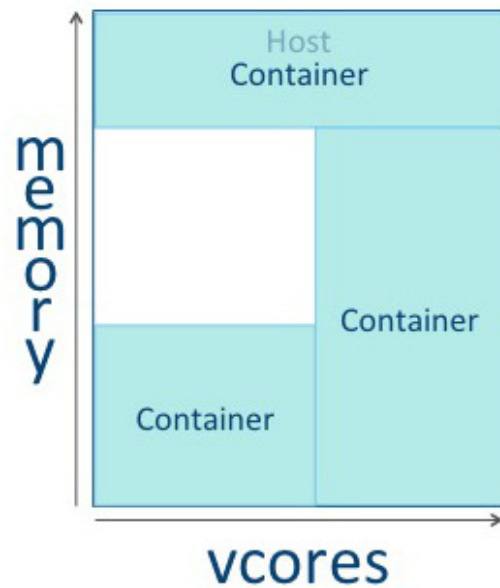
Tuning YARN consists primarily of optimally defining *containers* on your worker hosts. A container might be thought of as a rectangular graph consisting of memory and vcores. Containers perform tasks.



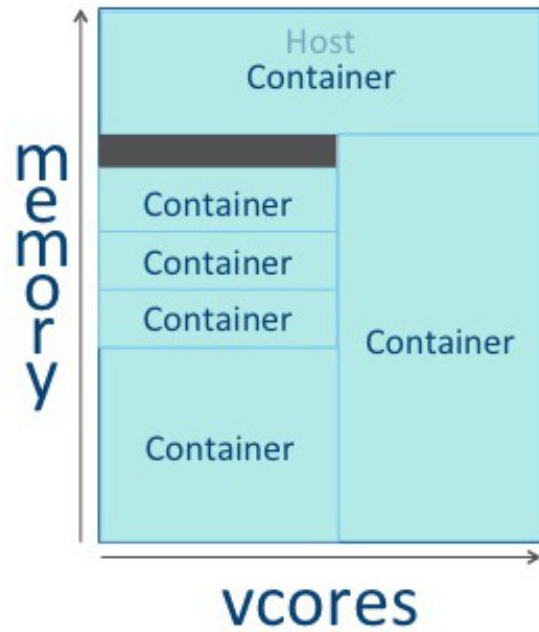
Some tasks use a great deal of memory, with minimal processing on a large volume of data.



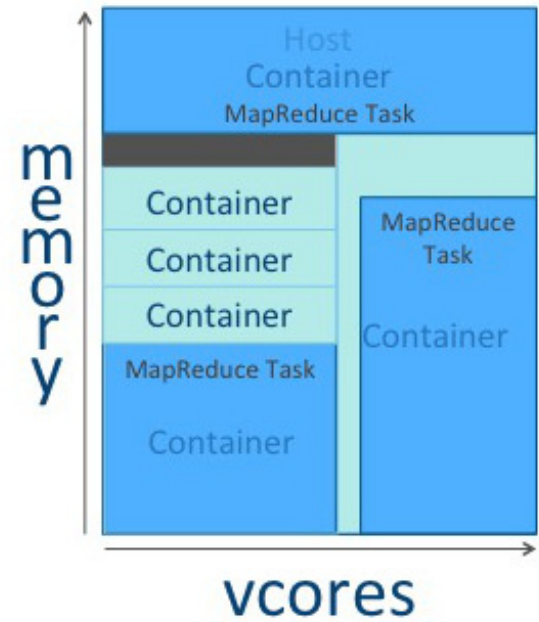
Other tasks require a great deal of processing power, but use less memory. For example, a Monte Carlo Simulation that evaluates many possible "what if?" scenarios uses a great deal of processing power on a relatively small dataset.



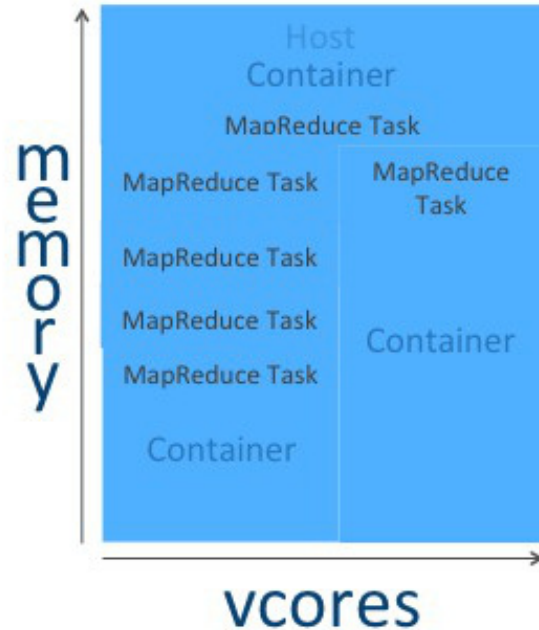
The YARN Resource Manager does its best to allocate memory and vcores to use all available resources in the most efficient way possible. Ideally there will be few or no resources left idle.



An *application* is a YARN client program that is made up of one or more tasks. Typically, a task uses all of the available resources in the container. A task is unable to consume more than its designated allocation, ensuring it cannot take over all of the host CPU cycles or exceed its memory allotment.



The goal of tuning your YARN hosts is to optimize your use of vcores and memory by configuring your containers to use all available resources beyond those required for overhead and other services.



There are three phases to YARN tuning. The phases correspond to the tabs in the [YARN tuning spreadsheet](#).

1. Cluster configuration, where you configure your hosts.
2. YARN configuration, where you quantify memory and vcores.
3. MapReduce configuration, where you allocate minimum and maximum resources for specific Map and Reduce tasks.

There are many configurable properties for YARN and MapReduce. You can see the exhaustive list at [Cloudera Manager Configuration Properties](#). The YARN tuning spreadsheet lists the essential subset of these properties that are most likely to improve performance for common MapReduce applications.

Cluster Configuration

In the Cluster Configuration tab, you define the worker host configuration and cluster size for your YARN implementation.

Step 1: Worker Host Configuration

Step 1 is to define the configuration for a single worker host computer in your cluster.

STEP 1: Worker Host Configuration		
Enter your likely machine configuration in the input boxes below. If you are uncertain what machines you plan on buying, put in some minimum values that will suit what you expect to buy. Last updated early 2016.		
Host Components	Quantity	Description
RAM	256	Gigabytes
CPU	48	8 CPUs: 6 cores, 3.5 GHz, 15MB cache
HDD (Hard Disk Drive)	36	12x3TB SATA III Hard Drives in JBOD Configuration
Ethernet	2	1 Gigabit Ethernet

As with any system, the more memory and CPU resources available, the faster the cluster can process large amounts of data. A machine with 8 CPUs, each with 6 cores, will give you 48 vcores per host.

Performance Management

3TB hard drives in a 2-unit server installation with 12 available slots in JBOD (Just a Bunch Of Disks) configuration is a reasonable balance of performance and pricing at the time the spreadsheet was created. Storage continues to get cheaper over time, so you might be able to get more for your money with 4TB disks. Larger disks are expensive, and not required for all use cases.

Two 1-Gigabit ethernet ports provide sufficient throughput at the time the spreadsheet was published, but 10-Gigabit ethernet ports are an option where price is of less concern than speed.

Step 2: Worker Host Planning

Step 2 is to allocate resources on each worker machine.

STEP 2: Worker Host Planning

Now that you have your base Host configuration from Step 1, use the table below to allocate resources, mainly CPU and memory, to the various software components that run on the host.

Service	Category	CPU (cores)	Memory (MB)	CM Static Service %	Notes
Operating System	Overhead	1	8192		N/A Most operating systems use 4-8GB minimum.
Cloudera Manager agent	Overhead	1	1024		N/A Allocate 1GB for Cloudera Manager agents, which track resource usage on a host.
Other services	Overhead	0	0		N/A Enter the required cores or memory for services not listed above.
HDFS DataNode	CDH	1	1024		4 Allocate 1GB for the HDFS DataNode.
Impala daemon	CDH	0	0		0 (Optional Service) Suggestion: Allocate at least 16GB memory when using Impala.
Hbase RegionServer	CDH	0	0		0 (Optional Service) Suggestion: Allocate no more than 12-16GB memory when using HBase Region Servers.
Solr Server	CDH	0	0		0 (Optional Service) Suggestion: Minimum 1GB for Solr server. More will be necessary depending on index sizes.
YARN NodeManager	CDH	1	1024		N/A Allocate 1GB for the YARN NodeManager.
Available Resources		44	250880		
Physical Cores to Vcores Multiplier		4			Set this ratio based on the expected number of concurrent threads per core. Use 1 for CPU intensive tasks up to 4 for standard I/O bound tasks.
YARN Available Vcores		176			This value will be used in STEP 4 for YARN Configuration
YARN Available Memory			250880		This value will be used in STEP 4 for YARN Configuration

Start with at least 8 GB for your operating system, and another GB for Cloudera Manager. If you know of services outside of CDH that require additional resources, add those numbers under Other Services.

You need the HDFS DataNode, using a minimum of 1 core and about 1 Gigabyte of memory. The same requirements apply to the YARN NodeManager.

The spreadsheet lists three optional services. For Impala, allocate at least 16 Gigabytes for the daemon. HBase RegionServer requires 12-16 GB of memory, but no more than that. Solr Server requires a minimum of 1GB of memory.

Whatever is left, in theory, is available for YARN applications (Spark and MapReduce). In this example, there are 44 CPU cores available. Set the multiplier for vcores you want on each physical core to calculate the total available vcores.

Step 3: Cluster Size

Having defined the specifications for each host in your cluster, enter the number of worker hosts needed to support your business case. 10 is a reasonable minimum number in order to see the benefits of parallel computing.

STEP 3: Cluster Size

Enter the number of nodes you have (or expect to have) in the cluster

		Quantity	
Number of Worker Hosts in the cluster		10	

YARN Configuration

On the YARN Configuration tab, you verify your available resources and set minimum and maximum limits for each container.

Steps 4 and 5: Verify Settings

Step 4 pulls forward the memory and vcore numbers from step 2. Step 5 shows the total memory and vcores for the cluster.

STEP 4: YARN Configuration on Cluster

These are the first set of configuration values for your cluster. You can set these values in YARN->Configuration in Cloudera Manager.

YARN Configuration Property	Value	
yarn.nodemanager.resource.cpu-vcores	176	Copied from STEP 2 "Available Resources"
yarn.nodemanager.resource.memory-mb	250880	Copied from STEP 2 "Available Resources"

STEP 5: Verify YARN Settings on Cluster

Go to the Resource Manager Web UI (usually <http://<ResourceManagerIP>:8088/> and verify the "Memory Total" and "Vcores Total" matches the values above. If your machine has no bad nodes, then the numbers should match exactly.

Resource Manager Property to Check	Value	Note
Expected Value for "Vcores Total"	1760	Calculated from STEP 2 "YARN Available Vcores" and STEP 3
Expected Value for "Memory Total" (in GB)	2450	Calculated from STEP 2 "YARN Available Memory" and STEP 3

Step 6: Verify Container Settings on Cluster

In step 6, you have the opportunity to change the 4 values that impact the size of your containers.

The minimum number of vcores should be 1. When additional vcores are required, adding 1 at a time should result in the most efficient allocation. Set the maximum number of vcore reservations for a container to ensure that no single task consumes all available resources.

Set the minimum and maximum reservations for memory. The increment should be the smallest amount that can have an impact on performance. Here, the minimum is approximately 1 GB, maximum is approximately 8 GB, and the increment is 512 MB.

STEP 6: Verify Container Settings on Cluster

In order to have YARN jobs run cleanly, you need to configure the container properties.

YARN Container Configuration Property (Vcores)	Value	Description
yarn.scheduler.minimum-allocation-vcores	1	Minimum vcore reservation for a container
yarn.scheduler.maximum-allocation-vcores	32	Maximum vcore reservation for a container
yarn.scheduler.increment-allocation-vcores	1	Vcore allocations must be a multiple of this value
YARN Container Configuration Property (Memory)	Value	
yarn.scheduler.minimum-allocation-mb	1024	Minimum memory reservation for a container
yarn.scheduler.maximum-allocation-mb	8192	Maximum memory reservation for a container
yarn.scheduler.increment-allocation-mb	512	Memory allocations must be a multiple of this value

Step 6A: Cluster Container Capacity

Step 6A lets you validate the minimum and maximum number of containers in your cluster, based on the numbers you entered.

Step 6A: Cluster Container Capacity

This section will tell you the capacity of your cluster (in terms of containers).

Cluster Container Estimates	Value
Largest number of containers, based on memory configuration	2450
Smallest number of containers, based on memory configuration	306
Largest number of containers, based on vcore configuration	1760
Smallest number of containers, based on vcore configuration	55

Step 6B: Container Sanity Checking

Step 6B lets you see at a glance whether you have over-allocated resources.

STEP 6B: Container Sanity Checking

This section will do some basic checking of your container parameters in STEP 6 against the hosts.

Sanity Check	Check Status	Description
Vcore Max >= Vcore Min	GOOD	yarn.scheduler.maximum-allocation-vcores must be greater than or equal to yarn.scheduler.minimum-allocation-vcores
Memory Max >= Memory Min	GOOD	yarn.scheduler.maximum-allocation-mb must be greater than or equal to yarn.scheduler.minimum-allocation-mb
VCoreMin <= HostsVCores	GOOD	yarn.scheduler.minimum-allocation-vcores must be less than or equal to the yarn.nodemanager.resource.cpu-vcores

MapReduce Configuration

On the MapReduce Configuration tab, you can plan for increased task-specific memory capacity.

Step 7: MapReduce Configuration

You can increase the memory allocation for the Application Master, map tasks, and reduce tasks. The minimum vcore allocation for any task is always 1. The Spill/Sort memory allocation of 256 should be sufficient, and should be (rarely) increased if you identify frequent spills to disk are hurting job performance.

STEP 7: MapReduce Configuration

Property	Property Type	Component	Value	Description
yarn.app.mapreduce.am.resource.cpu-vcores	Config	Application Master	1	AM container vcore reservation
yarn.app.mapreduce.am.resource.mb	Config	Application Master	1024	AM container memory reservation
mapreduce.map.cpu.vcores	Config	Map Task	1	Map task vcore reservation
mapreduce.map.memory.mb	Config	Map Task	1024	Map task memory reservation
mapreduce.reduce.cpu.vcores	Config	Reduce Task	1	Reduce task vcore reservation
mapreduce.reduce.memory.mb	Config	Reduce Task	1024	Reduce task memory reservation
mapreduce.task.io.sort.mb	Config	Spill/Sort (Map Task)	256	Spill/Sort memory reservation

Step 7A: MapReduce Sanity Checking

Step 7A lets you verify at a glance that all of your minimum and maximum resource allocations are within the parameters you set.

STEP 7A: MapReduce Sanity Checking

Sanity check MapReduce settings against container minimum/maximum properties.

Application Master Sanity Checks	Value	Description
yarn.app.mapreduce.am.resource.cpu-vcores >= container min	GOOD	Make sure ApplicationMaster vcore request fits within container limits
yarn.app.mapreduce.am.resource.cpu-vcores <= container max	GOOD	Ditto
yarn.app.mapreduce.am.resource.mb >= container min	GOOD	Make sure ApplicationMaster memory request fits within container limits
yarn.app.mapreduce.am.resource.mb <= container max	GOOD	Ditto
Map Task Sanity Checks	Value	Description
mapreduce.map.cpu.vcores >= container min	GOOD	Make sure Map Task vcore request fits within container limits
mapreduce.map.cpu.vcores <= container max	GOOD	Ditto
mapreduce.map.cpu.memory.mb >= container min	GOOD	Make sure Map Task memory request fits within container limits
mapreduce.map.cpu.memory.mb <= container max	GOOD	Ditto
Reduce Task Sanity Checks	Value	Description
mapreduce.reduce.cpu.vcores >= container min	GOOD	Make sure Reduce Task vcore request fits within container limits
mapreduce.reduce.cpu.vcores <= container max	GOOD	Ditto
mapreduce.reduce.cpu.memory.mb >= container min	GOOD	Make sure Reduce Task memory request fits within container limits
mapreduce.reduce.cpu.memory.mb <= container max	GOOD	Ditto

Configuring Your Cluster In Cloudera Manager

When you are satisfied with the cluster configuration estimates, use the values in the spreadsheet to set the corresponding properties in Cloudera Manager. For more information, see [Modifying Configuration Properties](#) on page 10

Table 11: Cloudera Manager Property Correspondence

Step	YARN/MapReduce Property	Cloudera Manager Equivalent
4	yarn.nodemanager.resource.cpu-vcores	Container Virtual CPU Cores
4	yarn.nodemanager.resource.memory-mb	Container Memory
6	yarn.scheduler.minimum-allocation-vcores	Container Virtual CPU Cores Minimum
6	yarn.scheduler.maximum-allocation-vcores	Container Virtual CPU Cores Maximum
6	yarn.scheduler.increment-allocation-vcores	Container Virtual CPU Cores Increment
6	yarn.scheduler.minimum-allocation-mb	Container Memory Minimum
6	yarn.scheduler.maximum-allocation-mb	Container Memory Maximum
6	yarn.scheduler.increment-allocation-mb	Container Memory Increment
7	yarn.app.mapreduce.am.resource.cpu-vcores	ApplicationMaster Virtual CPU Cores
7	yarn.app.mapreduce.am.resource.mb	ApplicationMaster Memory
7	mapreduce.map.cpu.vcores	Map Task CPU Virtual Cores
7	mapreduce.map.memory.mb	Map Task Memory
7	mapreduce.reduce.cpu.vcores	Reduce Task CPU Virtual Cores
7	mapreduce.reduce.memory.mb	Reduce Task Memory
7	mapreduce.task.io.sort.mb	I/O Sort Memory

High Availability

This guide is for Apache Hadoop system administrators who want to enable continuous availability by configuring clusters without single points of failure.

HDFS High Availability

This section provides an overview of the HDFS high availability (HA) feature and how to configure and manage an HA HDFS cluster.

Introduction to HDFS High Availability

This section assumes that the reader has a general understanding of components and node types in an HDFS cluster. For details, see the [Apache HDFS Architecture Guide](#).

Background

In a standard configuration, the NameNode is a single point of failure (SPOF) in an HDFS cluster. Each cluster has a single NameNode, and if that machine or process became unavailable, the cluster as a whole is unavailable until the NameNode is either restarted or brought up on a new host. The Secondary NameNode does not provide failover capability.

The standard configuration reduces the total availability of an HDFS cluster in two major ways:

- In the case of an unplanned event such as a host crash, the cluster is unavailable until an operator restarts the NameNode.
- Planned maintenance events such as software or hardware upgrades on the NameNode machine result in periods of cluster downtime.

HDFS HA addresses the above problems by providing the option of running two NameNodes in the same cluster, in an active/passive configuration. These are referred to as the active NameNode and the standby NameNode. Unlike the Secondary NameNode, the standby NameNode is hot standby, allowing a fast failover to a new NameNode in the case that a machine crashes, or a graceful administrator-initiated failover for the purpose of planned maintenance. You cannot have more than two NameNodes.

Implementation

Cloudera Manager 5 and CDH 5 support Quorum-based Storage to implement HA.

Quorum-based Storage

Quorum-based Storage refers to the HA implementation that uses a Quorum Journal Manager (QJM).

In order for the standby NameNode to keep its state synchronized with the active NameNode in this implementation, both nodes communicate with a group of separate daemons called JournalNodes. When any namespace modification is performed by the active NameNode, it durably logs a record of the modification to a majority of these JournalNodes. The standby NameNode is capable of reading the edits from the JournalNodes, and is constantly watching them for changes to the edit log. As the standby Node sees the edits, it applies them to its own namespace. In the event of a failover, the standby will ensure that it has read all of the edits from the JournalNodes before promoting itself to the active state. This ensures that the namespace state is fully synchronized before a failover occurs.

In order to provide a fast failover, it is also necessary that the standby NameNode has up-to-date information regarding the location of blocks in the cluster. In order to achieve this, the DataNodes are configured with the location of both NameNodes, and they send block location information and heartbeats to both.

It is vital for the correct operation of an HA cluster that only one of the NameNodes be active at a time. Otherwise, the namespace state would quickly diverge between the two, risking data loss or other incorrect results. In order to ensure this property and prevent the so-called "split-brain scenario," the JournalNodes will only ever allow a single

NameNode to be a writer at a time. During a failover, the NameNode which is to become active will simply take over the role of writing to the JournalNodes, which will effectively prevent the other NameNode from continuing in the active state, allowing the new active NameNode to safely proceed with failover.



Note: Because of this, fencing is not required, but it is still useful; see [Enabling HDFS HA](#) on page 288.

Automatic Failover

Automatic failover relies on two additional components in an HDFS: a ZooKeeper quorum, and the `ZKFailoverController` process (abbreviated as ZKFC). In Cloudera Manager, the ZKFC process maps to the HDFS Failover Controller role.

Apache ZooKeeper is a highly available service for maintaining small amounts of coordination data, notifying clients of changes in that data, and monitoring clients for failures. The implementation of HDFS automatic failover relies on ZooKeeper for the following functions:

- **Failure detection** - each of the NameNode machines in the cluster maintains a persistent session in ZooKeeper. If the machine crashes, the ZooKeeper session will expire, notifying the other NameNode that a failover should be triggered.
- **Active NameNode election** - ZooKeeper provides a simple mechanism to exclusively elect a node as active. If the current active NameNode crashes, another node can take a special exclusive lock in ZooKeeper indicating that it should become the next active NameNode.

The `ZKFailoverController` (ZKFC) is a ZooKeeper client that also monitors and manages the state of the NameNode. Each of the hosts that run a NameNode also run a ZKFC. The ZKFC is responsible for:

- **Health monitoring** - the ZKFC contacts its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds promptly with a healthy status, the ZKFC considers the NameNode healthy. If the NameNode has crashed, frozen, or otherwise entered an unhealthy state, the health monitor marks it as unhealthy.
- **ZooKeeper session management** - when the local NameNode is healthy, the ZKFC holds a session open in ZooKeeper. If the local NameNode is active, it also holds a special lock `znode`. This lock uses ZooKeeper's support for "ephemeral" nodes; if the session expires, the lock node is automatically deleted.
- **ZooKeeper-based election** - if the local NameNode is healthy, and the ZKFC sees that no other NameNode currently holds the lock `znode`, it will itself try to acquire the lock. If it succeeds, then it has "won the election", and is responsible for running a failover to make its local NameNode active. The failover process is similar to the manual failover described above: first, the previous active is fenced if necessary, and then the local NameNode transitions to active state.

General Questions about HDFS HA

What does the message "Operation category READ/WRITE is not supported in state standby" mean?

In an HA-enabled cluster, DFS clients cannot know in advance which NameNode is active at a given time. So when a client contacts a NameNode and it happens to be the standby, the READ or WRITE operation will be refused and this message is logged. The client will then automatically contact the other NameNode and try the operation again. As long as there is one active and one standby NameNode in the cluster, this message can be safely ignored.

If an application is configured to contact only one NameNode always, this message indicates that the application is failing to perform any read/write operation. In such situations, the application would need to be modified to use the HA configuration for the cluster. The Jira [HDFS-3447](#) deals with lowering the severity of this message (and similar ones) to DEBUG so as to reduce noise in the logs but is unresolved as of October 2018.

Configuring Hardware for HDFS HA

In order to deploy an HA cluster using Quorum-based Storage, you should prepare the following:

- NameNode machines - These are the machines on which you run the Active and Standby NameNodes. They should have equivalent hardware to each other, and equivalent hardware to what would be used in a non-HA cluster.
- JournalNode machines - These are the machines on which you run the JournalNodes. Cloudera recommends that you deploy the JournalNode daemons on the "master" host or hosts (NameNode, Standby NameNode, JobTracker, and so on) so the JournalNodes' local directories can use the reliable local storage on those machines.
- If co-located on the same machine, each JournalNode process and each NameNode process should have its own dedicated disk. You should not use SAN or NAS storage for these directories.
- There must be at least three JournalNode daemons, since edit log modifications must be written to a majority of JournalNodes. This will allow the system to tolerate the failure of a single machine. You can also run more than three JournalNodes, but in order to actually increase the number of failures the system can tolerate, you should run an odd number of JournalNodes, (three, five, seven, and so on). Note that when running with N JournalNodes, the system can tolerate at most $(N - 1) / 2$ failures and continue to function normally. If the requisite quorum is not available, the NameNode will not format or start, and you will see an error similar to this:

```
12/10/01 17:34:18 WARN namenode.FSEditLog: Unable to determine input streams from QJM
to [10.0.1.10:8485, 10.0.1.10:8486, 10.0.1.10:8487]. Skipping.
java.io.IOException: Timed out waiting 20000ms for a quorum of nodes to respond.
```



Note: In an HA cluster, the Standby NameNode also performs checkpoints of the namespace state, and thus it is not necessary to run a Secondary NameNode, CheckpointNode, or BackupNode in an HA cluster. In fact, to do so would be an error. If you are reconfiguring a non-HA-enabled HDFS cluster to be HA-enabled, you can reuse the hardware which you had previously dedicated to the Secondary NameNode.

Enabling HDFS HA

An HDFS high availability (HA) cluster uses two NameNodes—an active NameNode and a standby NameNode. Only one NameNode can be active at any point in time. HDFS HA depends on maintaining a log of all namespace modifications in a location available to both NameNodes, so that in the event of a failure, the standby NameNode has up-to-date information about the edits and location of blocks in the cluster.



Important: Enabling and disabling HA causes a service outage for the HDFS service and *all services* that depend on HDFS. Before enabling or disabling HA, ensure that there are no jobs running on your cluster.

Enabling HDFS HA Using Cloudera Manager

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

You can use Cloudera Manager to configure your CDH 5 cluster for HDFS HA and automatic failover. In Cloudera Manager 5, HA is implemented using Quorum-based storage. Quorum-based storage relies upon a set of JournalNodes, each of which maintains a local edits directory that logs the modifications to the namespace metadata. Enabling HA enables automatic failover as part of the same command.



Important:

- Enabling or disabling HA causes the previous monitoring history to become unavailable.
- Some parameters will be automatically set as follows once you have [enabled JobTracker HA](#). If you want to change the value from the default for these parameters, use an advanced configuration snippet.

```
- mapred.jobtracker.restart.recover: true
- mapred.job.tracker.persist.jobstatus.active: true
- mapred.ha.automatic-failover.enabled: true
- mapred.ha.fencing.methods: shell(/bin/true)
```


Enabling High Availability and Automatic Failover

The **Enable High Availability** workflow leads you through adding a second (standby) NameNode and configuring JournalNodes.

1. Go to the HDFS service.
 2. Select **Actions > Enable High Availability**. A screen showing the hosts that are eligible to run a standby NameNode and the JournalNodes displays.
 - a. Specify a name for the nameservice or accept the default name **nameservice1** and click **Continue**.
 - b. In the **NameNode Hosts** field, click **Select a host**. The host selection dialog box displays.
 - c. Check the checkbox next to the hosts where you want the standby NameNode to be set up and click **OK**. The standby NameNode cannot be on the same host as the active NameNode, and the host that is chosen should have the same hardware configuration (RAM, disk space, number of cores, and so on) as the active NameNode.
 - d. In the **JournalNode Hosts** field, click **Select hosts**. The host selection dialog box displays.
 - e. Check the checkboxes next to an odd number of hosts (a minimum of three) to act as JournalNodes and click **OK**. JournalNodes should be hosted on hosts with similar hardware specification as the NameNodes. Cloudera recommends that you put a JournalNode each on the same hosts as the active and standby NameNodes, and the third JournalNode on similar hardware, such as the JobTracker.
 - f. Click **Continue**.
 - g. In the **JournalNode Edits Directory** property, enter a directory location for the JournalNode edits directory into the fields for each JournalNode host.
 - You may enter only one directory for each JournalNode. The paths do not need to be the same on every JournalNode.
 - The directories you specify should be empty, and must have the appropriate permissions.
 - h. **Extra Options:** Decide whether Cloudera Manager should clear existing data in ZooKeeper, standby NameNode, and JournalNodes. If the directories are not empty (for example, you are re-enabling a previous HA configuration), Cloudera Manager will not automatically delete the contents—you can select to delete the contents by keeping the default checkbox selection. The recommended default is to clear the directories. If you choose not to do so, the data should be in sync across the edits directories of the JournalNodes and should have the same version data as the NameNodes.
 - i. Click **Continue**.
- Cloudera Manager executes a set of commands that will stop the dependent services, delete, create, and configure roles and directories as appropriate, create a nameservice and failover controller, and restart the dependent services and deploy the new client configuration.
3. If you want to use other services in a cluster with HA configured, follow the procedures in [Configuring Other CDH Components to Use HDFS HA](#) on page 302.



Important: If you change the NameNode Service RPC Port (`dfs.namenode.servicerpc-address`) while automatic failover is enabled, this will cause a mismatch between the NameNode address saved in the ZooKeeper `/hadoop-ha` znode and the NameNode address that the Failover Controller is configured with. This will prevent the Failover Controllers from restarting. If you need to change the NameNode Service RPC Port after Auto Failover has been enabled, you must do the following to re-initialize the znode:

1. Stop the HDFS service.
2. Configure the service RPC port:
 - a. Go to the HDFS service.
 - b. Click the **Configuration** tab.
 - c. Select **Scope > NameNode**.
 - d. Select **Category > Ports and Addresses**.
 - e. Locate the **NameNode Service RPC Port** property or search for it by typing its name in the Search box.
 - f. Change the port value as needed.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

3. On a ZooKeeper server host, run `zookeeper-client`.
 - a. Execute the following to remove the configured nameservice. This example assumes the name of the nameservice is **nameservice1**. You can identify the nameservice from the **Federation and High Availability** section on the **HDFS Instances** tab:

```
rmr /hadoop-ha/nameservice1
```

4. Click the **Instances** tab.
5. Select **Actions > Initialize High Availability State in ZooKeeper**.
6. Start the HDFS service.

Manually Failing Over to the Standby NameNode

If you are running an HDFS service with HA enabled, you can manually cause the active NameNode to failover to the standby NameNode. This is useful for planned downtime—for hardware changes, configuration changes, or software upgrades of your primary host.

1. Go to the HDFS service.
2. Click the **Instances** tab.
3. Select **Actions > Manual Failover**. (This option does not appear if HA is not enabled for the cluster.)
4. From the pop-up, select the NameNode that should be made active, then click **Manual Failover**.



Note: For advanced use only: To force the selected NameNode to be active, irrespective of its state or the other NameNode state, set the **Force Failover** checkbox. You should choose this option only if automatic failover is *not enabled*. Forcing a failover will first attempt to failover the selected NameNode to active mode and the other NameNode to standby mode. It will do so even if the selected NameNode is in safe mode. If this fails, it will proceed to transition the selected NameNode to active mode. To avoid having two NameNodes be active, use this only if the other NameNode is either definitely stopped, or can be transitioned to standby mode by the first failover step.

5. When all the steps have been completed, click **Finish**.

Cloudera Manager transitions the NameNode you selected to be the active NameNode, and the other NameNode to be the standby NameNode. HDFS should *never* have two active NameNodes.

Fencing Methods

In order to ensure that only one NameNode is active at a time, a fencing method is required for the shared edits directory. During a failover, the fencing method is responsible for ensuring that the previous active NameNode no longer has access to the shared edits directory, so that the new active NameNode can safely proceed writing to it.

By default, Cloudera Manager configures HDFS to use a shell fencing method (`shell(/cloudera_manager_agent_fencer.py)`) that takes advantage of the Cloudera Manager Agent. However, you can configure HDFS to use the `sshfence` method, or you can add your own shell fencing scripts, instead of or in addition to the one Cloudera Manager provides.

The fencing parameters are found in the **Service-Wide > High Availability** category under the configuration properties for your HDFS service.

For details of the fencing methods supplied with CDH 5, and how fencing is configured, see [Fencing Configuration](#) on page 294.

Enabling HDFS HA Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

This section describes the software configuration required for HDFS HA in CDH 5 and explains how to set configuration properties and use the command line to deploy HDFS HA.

Configuring Software for HDFS HA

Configuration Overview

As with HDFS Federation configuration, HA configuration is backward compatible and allows existing single NameNode configurations to work without change. The new configuration is designed such that all the nodes in the cluster can have the same configuration without the need for deploying different configuration files to different machines based on the type of the node.

HA clusters reuse the **Nameservice ID** to identify a single HDFS instance that may consist of multiple HA NameNodes. In addition, there is a new abstraction called **NameNode ID**. Each distinct NameNode in the cluster has a different NameNode ID. To support a single configuration file for all of the NameNodes, the relevant configuration parameters include the Nameservice ID as well as the NameNode ID.

Changes to Existing Configuration Parameters

The following configuration parameter has changed for YARN implementations:

`fs.defaultFS` - formerly `fs.default.name`, the default path prefix used by the Hadoop FS client when none is given. (`fs.default.name` is deprecated for YARN implementations, but will still work.)

Optionally, you can configure the default path for Hadoop clients to use the HA-enabled logical URI. For example, if you use `mycluster` as the Nameservice ID as shown below, this will be the value of the authority portion of all of your HDFS paths. You can configure the default path in your `core-site.xml` file:

- For YARN:

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://mycluster</value>
</property>
```

- For MRv1:

```
<property>
  <name>fs.default.name</name>
  <value>hdfs://mycluster</value>
</property>
```

New Configuration Parameters

To configure HA NameNodes, you must add several configuration options to your `hdfs-site.xml` configuration file.

The order in which you set these configurations is unimportant, but the values you choose for `dfs.nameservices` and `dfs.ha.namenodes.[Nameservice ID]` will determine the keys of those that follow. This means that you should decide on these values before setting the rest of the configuration options.

Configure `dfs.nameservices`

`dfs.nameservices` - the logical name for this new nameservice

Choose a logical name for this nameservice, for example `mycluster`, and use this logical name for the value of this configuration option. The name you choose is arbitrary. It will be used both for configuration and as the authority component of absolute HDFS paths in the cluster.



Note: If you are also using HDFS Federation, this configuration setting should also include the list of other Nameservices, HA or otherwise, as a comma-separated list.

```
<property>
  <name>dfs.nameservices</name>
  <value>mycluster</value>
</property>
```

Configure `dfs.ha.namenodes.[nameservice ID]`

`dfs.ha.namenodes.[nameservice ID]` - unique identifiers for each NameNode in the nameservice

Configure a list of comma-separated NameNode IDs. This will be used by DataNodes to determine all the NameNodes in the cluster. For example, if you used `mycluster` as the NameService ID previously, and you wanted to use `nn1` and `nn2` as the individual IDs of the NameNodes, you would configure this as follows:

```
<property>
  <name>dfs.ha.namenodes.mycluster</name>
  <value>nn1,nn2</value>
</property>
```



Note: In this release, you can configure a maximum of two NameNodes per nameservice.

Configure `dfs.namenode.rpc-address.[nameservice ID]`

`dfs.namenode.rpc-address.[nameservice ID].[name node ID]` - the fully-qualified RPC address for each NameNode to listen on

For both of the previously-configured NameNode IDs, set the full address and RPC port of the NameNode process. Note that this results in two separate configuration options. For example:

```
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn1</name>
  <value>machine1.example.com:8020</value>
</property>
<property>
  <name>dfs.namenode.rpc-address.mycluster.nn2</name>
```

```
<value>machine2.example.com:8020</value>
</property>
```



Note: If necessary, you can similarly configure the `servicerpc-address` setting.

Configure `dfs.namenode.http-address.[nameservice ID]`

`dfs.namenode.http-address.[nameservice ID].[name node ID]` - the fully-qualified HTTP address for each NameNode to listen on

Similarly to `rpc-address` above, set the addresses for both NameNodes' HTTP servers to listen on. For example:

```
<property>
  <name>dfs.namenode.http-address.mycluster.nn1</name>
  <value>machine1.example.com:50070</value>
</property>
<property>
  <name>dfs.namenode.http-address.mycluster.nn2</name>
  <value>machine2.example.com:50070</value>
</property>
```



Note: If you have Hadoop Kerberos security features enabled, and you intend to use HSFTP, you should also set the `https-address` similarly for each NameNode.

Configure `dfs.namenode.shared.edits.dir`

`dfs.namenode.shared.edits.dir` - the location of the shared storage directory

Configure the addresses of the JournalNodes which provide the shared edits storage, written to by the Active NameNode and read by the Standby NameNode to stay up-to-date with all the file system changes the Active NameNode makes. Though you must specify several JournalNode addresses, **you should only configure one of these URIs**. The URI should be in the form:

```
qjournal://<host1:port1>;<host2:port2>;<host3:port3>/<journalId>
```

The Journal ID is a unique identifier for this nameservice, which allows a single set of JournalNodes to provide storage for multiple federated namesystems. Though it is not a requirement, it's a good idea to reuse the Nameservice ID for the journal identifier.

For example, if the JournalNodes for this cluster were running on the machines `node1.example.com`, `node2.example.com`, and `node3.example.com`, and the nameservice ID were `mycluster`, you would use the following as the value for this setting (the default port for the JournalNode is 8485):

```
<property>
  <name>dfs.namenode.shared.edits.dir</name>

  <value>qjournal://node1.example.com:8485;node2.example.com:8485;node3.example.com:8485/mycluster</value>
</property>
```

Configure `dfs.journalnode.edits.dir`

`dfs.journalnode.edits.dir` - the path where the JournalNode daemon will store its local state

On each JournalNode machine, configure the absolute path where the edits and other local state information used by the JournalNodes will be stored; use only a single path per JournalNode. (The other JournalNodes provide redundancy; you can also configure this directory on a locally-attached RAID-1 or RAID-10 array.)

For example:

```
<property>
  <name>dfs.journalnode.edits.dir</name>
  <value>/data/1/dfs/jn</value>
</property>
```

Now create the directory (if it doesn't already exist) and make sure its owner is `hdfs`, for example:

```
$ sudo mkdir -p /data/1/dfs/jn
$ sudo chown -R hdfs:hdfs /data/1/dfs/jn
```

Client Failover Configuration

`dfs.client.failover.proxy.provider.[nameservice ID]` - the Java class that HDFS clients use to contact the Active NameNode

Configure the name of the Java class which the DFS client will use to determine which NameNode is the current active, and therefore which NameNode is currently serving client requests. The only implementation which currently ships with Hadoop is the `ConfiguredFailoverProxyProvider`, so use this unless you are using a custom one. For example:

```
<property>
  <name>dfs.client.failover.proxy.provider.mycluster</name>
  <value>org.apache.hadoop.hdfs.server.namenode.ha.ConfiguredFailoverProxyProvider</value>
</property>
```

Fencing Configuration

`dfs.ha.fencing.methods` - a list of scripts or Java classes which will be used to fence the active NameNode during a failover

It is desirable for correctness of the system that only one NameNode be in the active state at any given time.



Important: When you use Quorum-based Storage, only one NameNode will ever be allowed to write to the JournalNodes, so there is no potential for corrupting the file system metadata in a "split-brain" scenario. But when a failover occurs, it is still possible that the previously active NameNode could serve read requests to clients - and these requests may be out of date - until that NameNode shuts down when it tries to write to the JournalNodes. For this reason, it is still desirable to configure some fencing methods even when using Quorum-based Storage.

To improve the availability of the system in the event the fencing mechanisms fail, it is advisable to configure a fencing method which is guaranteed to return success as the last fencing method in the list.



Note: If you choose to use no actual fencing methods, you still must configure something for this setting, for example `shell(/bin/true)`.

The fencing methods used during a failover are configured as a carriage-return-separated list, and these will be attempted in order until one of them indicates that fencing has succeeded.

There are two fencing methods which ship with Hadoop:

- [sshfence](#)
- [shell](#)

For information on implementing your own custom fencing method, see the `org.apache.hadoop.ha.NodeFencer` class.

Configuring the sshfence fencing method

`sshfence` - SSH to the active NameNode and kill the process

The `sshfence` option uses SSH to connect to the target node and uses `fuser` to kill the process listening on the service's TCP port. In order for this fencing option to work, it must be able to SSH to the target node without providing a passphrase. Thus, you must also configure the `dfs.ha.fencing.ssh.private-key-files` option, which is a comma-separated list of SSH private key files.



Important: The files must be accessible to the user running the NameNode processes (typically the `hdfs` user on the NameNode hosts).

For example:

```
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>sshfence</value>
</property>

<property>
  <name>dfs.ha.fencing.ssh.private-key-files</name>
  <value>/home/exampleuser/.ssh/id_rsa</value>
</property>
```

Optionally, you can configure a non-standard username or port to perform the SSH as shown below. You can also configure a timeout, in milliseconds, for the SSH, after which this fencing method will be considered to have failed:

```
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>sshfence([[username][:port]])</value>
</property>
<property>
  <name>dfs.ha.fencing.ssh.connect-timeout</name>
  <value>30000</value>
  <description>
    SSH connection timeout, in milliseconds, to use with the builtin
    sshfence fencer.
  </description>
</property>
```

Configuring the shell fencing method

`shell` - run an arbitrary shell command to fence the active NameNode

The shell fencing method runs an arbitrary shell command, which you can configure as shown below:

```
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>shell(/path/to/my/script.sh arg1 arg2 ...)</value>
</property>
```

The string between '(' and ')' is passed directly to a `bash` shell and cannot include any closing parentheses.

When executed, the first argument to the configured script will be the address of the NameNode to be fenced, followed by all arguments specified in the configuration.

The shell command will be run with an environment set up to contain all of the current Hadoop configuration variables, with the '_' character replacing any '.' characters in the configuration keys. The configuration used has already had any NameNode-specific configurations promoted to their generic forms - for example `dfs_namenode_rpc_address` will contain the RPC address of the target node, even though the configuration may specify that variable as `dfs.namenode.rpc-address.ns1.nn1`.

The following variables referring to the target node to be fenced are also available:

Variable	Description
<code>\$target_host</code>	Hostname of the node to be fenced

Variable	Description
\$target_port	IPC port of the node to be fenced
\$target_address	The above two variables, combined as <i>host:port</i>
\$target_nameserviceid	The nameservice ID of the NameNode to be fenced
\$target_namenodeid	The NameNode ID of the NameNode to be fenced

You can also use these environment variables as substitutions in the shell command itself. For example:

```
<property>
  <name>dfs.ha.fencing.methods</name>
  <value>shell(/path/to/my/script.sh --nameservice=$target_nameserviceid
target_host:$target_port)</value>
</property>
```

If the shell command returns an exit code of 0, the fencing is determined to be successful. If it returns any other exit code, the fencing was not successful and the next fencing method in the list will be attempted.



Note: This fencing method does not implement any timeout. If timeouts are necessary, they should be implemented in the shell script itself (for example, by forking a subshell to kill its parent in some number of seconds).

Automatic Failover Configuration

The above sections describe how to configure manual failover. In that mode, the system will not automatically trigger a failover from the active to the standby NameNode, even if the active node has failed. This section describes how to configure and deploy automatic failover.

Component Overview

Automatic failover adds two new components to an HDFS deployment: a ZooKeeper quorum, and the `ZKFailoverController` process (abbreviated as ZKFC).

Apache ZooKeeper is a highly available service for maintaining small amounts of coordination data, notifying clients of changes in that data, and monitoring clients for failures. The implementation of automatic HDFS failover relies on ZooKeeper for the following things:

- **Failure detection** - each of the NameNode machines in the cluster maintains a persistent session in ZooKeeper. If the machine crashes, the ZooKeeper session will expire, notifying the other NameNode that a failover should be triggered.
- **Active NameNode election** - ZooKeeper provides a simple mechanism to exclusively elect a node as active. If the current active NameNode crashes, another node can take a special exclusive lock in ZooKeeper indicating that it should become the next active NameNode.

The `ZKFailoverController` (ZKFC) is a new component - a ZooKeeper client which also monitors and manages the state of the NameNode. Each of the machines which runs a NameNode also runs a ZKFC, and that ZKFC is responsible for:

- **Health monitoring** - the ZKFC pings its local NameNode on a periodic basis with a health-check command. So long as the NameNode responds promptly with a healthy status, the ZKFC considers the node healthy. If the node has crashed, frozen, or otherwise entered an unhealthy state, the health monitor will mark it as unhealthy.
- **ZooKeeper session management** - when the local NameNode is healthy, the ZKFC holds a session open in ZooKeeper. If the local NameNode is active, it also holds a special lock `znode`. This lock uses ZooKeeper's support for "ephemeral" nodes; if the session expires, the lock node will be automatically deleted.
- **ZooKeeper-based election** - if the local NameNode is healthy, and the ZKFC sees that no other node currently holds the lock `znode`, it will itself try to acquire the lock. If it succeeds, then it has "won the election", and is responsible for running a failover to make its local NameNode active. The failover process is similar to the manual

failover described above: first, the previous active is fenced if necessary, and then the local NameNode transitions to active state.

Deploying ZooKeeper

In a typical deployment, ZooKeeper daemons are configured to run on three or five nodes. Since ZooKeeper itself has light resource requirements, it is acceptable to collocate the ZooKeeper nodes on the same hardware as the HDFS NameNode and Standby Node. Operators using MapReduce v2 (MRv2) often choose to deploy the third ZooKeeper process on the same node as the YARN ResourceManager. It is advisable to configure the ZooKeeper nodes to store their data on separate disk drives from the HDFS metadata for best performance and isolation.

See the [ZooKeeper documentation](#) for instructions on how to set up a ZooKeeper ensemble. In the following sections we assume that you have set up a ZooKeeper cluster running on three or more nodes, and have verified its correct operation by connecting using the ZooKeeper command-line interface (CLI).

Configuring Automatic Failover



Note: Before you begin configuring automatic failover, you must shut down your cluster. It is not currently possible to transition from a manual failover setup to an automatic failover setup while the cluster is running.

Configuring automatic failover requires two additional configuration parameters. In your `hdfs-site.xml` file, add:

```
<property>
  <name>dfs.ha.automatic-failover.enabled</name>
  <value>true</value>
</property>
```

This specifies that the cluster should be set up for automatic failover. In your `core-site.xml` file, add:

```
<property>
  <name>ha.zookeeper.quorum</name>
  <value>zk1.example.com:2181,zk2.example.com:2181,zk3.example.com:2181</value>
</property>
```

This lists the host-port pairs running the ZooKeeper service.

As with the parameters described earlier in this document, these settings may be configured on a per-nameservice basis by suffixing the configuration key with the nameservice ID. For example, in a cluster with federation enabled, you can explicitly enable automatic failover for only one of the nameservices by setting `dfs.ha.automatic-failover.enabled.my-nameservice-id`.

There are several other configuration parameters which you can set to control the behavior of automatic failover, but they are not necessary for most installations. See the configuration section of the [Hadoop documentation](#) for details.

Initializing the HA state in ZooKeeper

After you have added the configuration keys, the next step is to initialize the required state in ZooKeeper. You can do so by running the following command from one of the NameNode hosts.



Note: The ZooKeeper ensemble must be running when you use this command; otherwise it will not work properly.

```
$ hdfs zkfc -formatZK
```

This will create a `znode` in ZooKeeper in which the automatic failover system stores its data.

Securing access to ZooKeeper

If you are running a secure cluster, you will probably want to ensure that the information stored in ZooKeeper is also secured. This prevents malicious clients from modifying the metadata in ZooKeeper or potentially triggering a false failover.

In order to secure the information in ZooKeeper, first add the following to your `core-site.xml` file:

```
<property>
  <name>ha.zookeeper.auth</name>
  <value>@/path/to/zk-auth.txt</value>
</property>
<property>
  <name>ha.zookeeper.acl</name>
  <value>@/path/to/zk-acl.txt</value>
</property>
```

Note the '@' character in these values – this specifies that the configurations are not inline, but rather point to a file on disk.

The first configured file specifies a list of ZooKeeper authentications, in the same format as used by the ZooKeeper CLI. For example, you may specify something like `digest:hdfs-zkfc:mypassword` where `hdfs-zkfc` is a unique username for ZooKeeper, and `mypassword` is some unique string used as a password.

Next, generate a ZooKeeper Access Control List (ACL) that corresponds to this authentication, using a command such as the following:

```
$ java -cp $ZK_HOME/lib/*:$ZK_HOME/zookeeper-3.4.2.jar
org.apache.zookeeper.server.auth.DigestAuthenticationProvider hdfs-zkfc:mypassword
output: hdfs-zkfc:mypassword->hdfs-zkfc:P/OQvnYyU/nF/mGYvB/xurX8dYs=
```

Copy and paste the section of this output after the '->' string into the file `zk-acls.txt`, prefixed by the string "digest:". For example:

```
digest:hdfs-zkfc:v1UvLnd8MlacseE80rDuu6ONESbM=:rwcda
```

To put these ACLs into effect, rerun the `zkfc -formatZK` command as described above.

After doing so, you can verify the ACLs from the ZooKeeper CLI as follows:

```
[zk: localhost:2181(CONNECTED) 1] getAcl /hadoop-ha
'digest, 'hdfs-zkfc:v1UvLnd8MlacseE80rDuu6ONESbM=
: cdrwa
```

Automatic Failover FAQ

Is it important that I start the ZKFC and NameNode daemons in any particular order?

No. On any given node you may start the ZKFC before or after its corresponding NameNode.

What additional monitoring should I put in place?

You should add monitoring on each host that runs a NameNode to ensure that the ZKFC remains running. In some types of ZooKeeper failures, for example, the ZKFC may unexpectedly exit, and should be restarted to ensure that the system is ready for automatic failover. Additionally, you should monitor each of the servers in the ZooKeeper quorum. If ZooKeeper crashes, automatic failover will not function.

What happens if ZooKeeper goes down?

If the ZooKeeper cluster crashes, no automatic failovers will be triggered. However, HDFS will continue to run without any impact. When ZooKeeper is restarted, HDFS will reconnect with no issues.

Can I designate one of my NameNodes as primary/preferred?

No. Currently, this is not supported. Whichever NameNode is started first will become active. You may choose to start the cluster in a specific order such that your preferred node starts first.

How can I initiate a manual failover when automatic failover is configured?

Even if automatic failover is configured, you can initiate a [manual failover](#). It will perform a coordinated failover.

Deploying HDFS High Availability

After you have set all of the necessary configuration options, you are ready to start the JournalNodes and the two HA NameNodes.



Important: Before you start: Make sure you have performed all the configuration and setup tasks described under [Configuring Hardware for HDFS HA](#) on page 287 and [Configuring Software for HDFS HA](#) on page 291, including initializing the HA state in ZooKeeper if you are deploying automatic failover.

Install and Start the JournalNodes

1. Install the JournalNode daemons on each of the machines where they will run.

To install JournalNode on Red Hat-compatible systems:

```
$ sudo yum install hadoop-hdfs-journalnode
```

To install JournalNode on Ubuntu and Debian systems:

```
$ sudo apt-get install hadoop-hdfs-journalnode
```

To install JournalNode on SLES systems:

```
$ sudo zypper install hadoop-hdfs-journalnode
```

2. Start the JournalNode daemons on each of the machines where they will run:

```
sudo service hadoop-hdfs-journalnode start
```

Wait for the daemons to start before formatting the primary NameNode (in a new cluster) and before starting the NameNodes (in all cases).

Format the NameNode (if new cluster)

If you are setting up a new HDFS cluster, format the NameNode you will use as your primary NameNode; see [Formatting the NameNode](#).



Important: Make sure the JournalNodes have started. Formatting will fail if you have configured the NameNode to communicate with the JournalNodes, but have not started the JournalNodes.

Initialize the Shared Edits directory (if converting existing non-HA cluster)

If you are converting a non-HA NameNode to HA, initialize the shared edits directory with the edits data from the local NameNode edits directories:

```
hdfs namenode -initializeSharedEdits
```

Start the NameNodes

1. Start the primary (formatted) NameNode:

```
$ sudo service hadoop-hdfs-namenode start
```

2. Start the standby NameNode:

```
$ sudo -u hdfs hdfs namenode -bootstrapStandby
$ sudo service hadoop-hdfs-namenode start
```



Note: If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) or `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

Starting the standby NameNode with the `-bootstrapStandby` option copies over the contents of the primary NameNode's metadata directories (including the namespace information and most recent checkpoint) to the standby NameNode. (The location of the directories containing the NameNode metadata is configured using the configuration options `dfs.namenode.name.dir` and `dfs.namenode.edits.dir`.)

You can visit each NameNode's web page by browsing to its configured HTTP address. Notice that next to the configured address is the HA state of the NameNode (either "Standby" or "Active".) Whenever an HA NameNode starts and automatic failover is not enabled, it is initially in the Standby state. If automatic failover is enabled the first NameNode that is started will become active.

Restart Services (if converting existing non-HA cluster)

If you are converting from a non-HA to an HA configuration, you need to restart the JobTracker and TaskTracker (for MRv1, if used), or ResourceManager, NodeManager, and JobHistory Server (for YARN), and the DataNodes:

On each DataNode:

```
$ sudo service hadoop-hdfs-datanode start
```

On each TaskTracker system (MRv1):

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

On the JobTracker system (MRv1):

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

Verify that the JobTracker and TaskTracker started properly:

```
sudo jps | grep Tracker
```

On the ResourceManager system (YARN):

```
$ sudo service hadoop-yarn-resourcemanager start
```

On each NodeManager system (YARN; typically the same ones where DataNode service runs):

```
$ sudo service hadoop-yarn-nodemanager start
```

On the MapReduce JobHistory Server system (YARN):

```
$ sudo service hadoop-mapreduce-historyserver start
```

Deploy Automatic Failover (if it is configured)

If you have configured automatic failover using the ZooKeeper FailoverController (ZKFC), you must install and start the `zkfc` daemon on each of the machines that runs a NameNode. Proceed as follows.

To install ZKFC on Red Hat-compatible systems:

```
$ sudo yum install hadoop-hdfs-zkfc
```

To install ZKFC on Ubuntu and Debian systems:

```
$ sudo apt-get install hadoop-hdfs-zkfc
```

To install ZKFC on SLES systems:

```
$ sudo zypper install hadoop-hdfs-zkfc
```

To start the zkfc daemon:

```
$ sudo service hadoop-hdfs-zkfc start
```

It is not important that you start the ZKFC and NameNode daemons in a particular order. On any given node you can start the ZKFC before or after its corresponding NameNode.

You should add monitoring on each host that runs a NameNode to ensure that the ZKFC remains running. In some types of ZooKeeper failures, for example, the ZKFC may unexpectedly exit, and should be restarted to ensure that the system is ready for automatic failover.

Additionally, you should monitor each of the servers in the ZooKeeper quorum. If ZooKeeper crashes, then automatic failover will not function. If the ZooKeeper cluster crashes, no automatic failovers will be triggered. However, HDFS will continue to run without any impact. When ZooKeeper is restarted, HDFS will reconnect with no issues.

Verifying Automatic Failover

After the initial deployment of a cluster with automatic failover enabled, you should test its operation. To do so, first locate the active NameNode. As mentioned above, you can tell which node is active by visiting the NameNode web interfaces.

Once you have located your active NameNode, you can cause a failure on that node. For example, you can use `kill -9 <pid of NN>` to simulate a JVM crash. Or you can power-cycle the machine or its network interface to simulate different kinds of outages. After you trigger the outage you want to test, the other NameNode should automatically become active within several seconds. The amount of time required to detect a failure and trigger a failover depends on the configuration of `ha.zookeeper.session-timeout.ms`, but defaults to 5 seconds.

If the test does not succeed, you may have a misconfiguration. Check the logs for the `zkfc` daemons as well as the NameNode daemons in order to further diagnose the issue.

Disabling and Redeploying HDFS HA

Disabling and Redeploying HDFS HA Using Cloudera Manager

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

1. Go to the HDFS service.
2. Select **Actions > Disable High Availability**.
3. Select the hosts for the NameNode and the SecondaryNameNode and click **Continue**.
4. Select the HDFS checkpoint directory and click **Continue**.
5. Confirm that you want to take this action.
6. [Update the Hive Metastore NameNode](#).

Cloudera Manager ensures that one NameNode is active, and saves the namespace. Then it stops the standby NameNode, creates a SecondaryNameNode, removes the standby NameNode role, and restarts all the HDFS services.

Disabling and Redeploying HDFS HA Using the Command Line

**Important:**

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

If you need to unconfigure HA and revert to using a single NameNode, either permanently or for upgrade or testing purposes, proceed as follows.



Important: Only [Quorum-based storage](#) is supported in CDH 5. If you already using Quorum-based storage, you *do not* need to unconfigure it to upgrade.

Step 1: Shut Down the Cluster

1. Shut down Hadoop services across your entire cluster. Do this from Cloudera Manager; or, if you are not using Cloudera Manager, run the following command on every host in your cluster:

```
$ for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done
```

2. Check each host to make sure that there are no processes running as the `hdfs`, `yarn`, `mapred` or `httpfs` users from root:

```
# ps -aef | grep java
```

Step 2: Unconfigure HA

1. Disable the software configuration.
 - If you are using Quorum-based storage and want to unconfigure it, unconfigure the HA properties described under [Enabling HDFS HA Using the Command Line](#) on page 291.
If you intend to redeploy HDFS HA later, comment out the HA properties rather than deleting them.
2. Move the NameNode metadata directories on the standby NameNode. The location of these directories is configured by `dfs.namenode.name.dir` and `dfs.namenode.edits.dir`. Move them to a backup location.

Step 3: Restart the Cluster

```
for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x start ; done
```

Redeploying HDFS High Availability

If you need to redeploy HA using Quorum-based storage after temporarily disabling it, proceed as follows:

1. Shut down the cluster as described in [Step 1: Shut Down the Cluster](#) on page 302.
2. Uncomment the properties you commented out in [Step 2: Unconfigure HA](#) on page 302.
3. Deploy HDFS HA, following the instructions under [Deploying HDFS High Availability](#) on page 299.

Configuring Other CDH Components to Use HDFS HA

You can use the HDFS high availability NameNodes with other components of CDH.

Configuring HBase to Use HDFS HA

Configuring HBase to Use HDFS HA Using the Command Line

To configure HBase to use HDFS HA, proceed as follows.

Shut Down the HBase Cluster

1. Stop the Thrift server and clients:

```
sudo service hbase-thrift stop
```

2. Stop the cluster by shutting down the Master and the RegionServers:

- Use the following command on the Master host:

```
sudo service hbase-master stop
```

- Use the following command on each host hosting a RegionServer:

```
sudo service hbase-regionserver stop
```

Configure hbase.rootdir

Change the distributed file system URI in `hbase-site.xml` to the name specified in the `dfs.nameservices` property in `hdfs-site.xml`. The clients must also have access to `hdfs-site.xml`'s `dfs.client.*` settings to properly use HA.

For example, suppose the HDFS HA property `dfs.nameservices` is set to `ha-nn` in `hdfs-site.xml`. To configure HBase to use the HA NameNodes, specify that same value as part of your `hbase-site.xml`'s `hbase.rootdir` value:

```
<!-- Configure HBase to use the HA NameNode nameservice -->
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://ha-nn/hbase</value>
</property>
```

Restart HBase

1. Start the HBase Master.
2. Start each of the HBase RegionServers.

HBase-HDFS HA Troubleshooting

Problem: HMasters fail to start.

Solution: Check for this error in the HMaster log:

```
2012-05-17 12:21:28,929 FATAL master.HMaster (HMaster.java:abort(1317)) - Unhandled
exception. Starting shutdown.
java.lang.IllegalArgumentException: java.net.UnknownHostException: ha-nn
    at
org.apache.hadoop.security.SecurityUtil.buildTokenService(SecurityUtil.java:431)
    at
org.apache.hadoop.hdfs.NameNodeProxies.createNonHAProxy(NameNodeProxies.java:161)
    at org.apache.hadoop.hdfs.NameNodeProxies.createProxy(NameNodeProxies.java:126)
    ...
```

If so, verify that Hadoop's `hdfs-site.xml` and `core-site.xml` files are in your `hbase/conf` directory. This may be necessary if you put your configurations in non-standard places.

Upgrading the Hive Metastore to Use HDFS HA

The Hive metastore can be configured to use HDFS high availability.

Upgrading the Hive Metastore to Use HDFS HA Using Cloudera Manager

1. Go to the Hive service.
2. Select **Actions > Stop**.



Note: You may want to stop the Hue and Impala services first, if present, as they depend on the Hive service.

Click **Stop** to confirm the command.

3. Back up the Hive metastore database.
4. Select **Actions > Update Hive Metastore NameNodes** and confirm the command.
5. Select **Actions > Start**.
6. Restart the Hue and Impala services if you stopped them prior to updating the metastore.

Upgrading the Hive Metastore to Use HDFS HA Using the Command Line

To configure the Hive metastore to use HDFS HA, change the records to reflect the location specified in the `dfs.nameservices` property, using the `Hive metatool` to obtain and change the locations.



Note: Before attempting to upgrade the Hive metastore to use HDFS HA, shut down the metastore and back it up to a persistent store.

If you are unsure which version of Avro SerDe is used, use both the `serdePropKey` and `tablePropKey` arguments. For example:

```
$ hive --service metatool -listFSRoot
hdfs://oldnamenode.com/user/hive/warehouse
$ metatool -updateLocation hdfs://nameservice1 hdfs://oldnamenode.com -tablePropKey
avro.schema.url
-serdePropKey schema.url
$ hive --service metatool -listFSRoot
hdfs://nameservice1/user/hive/warehouse
```

where:

- `hdfs://oldnamenode.com/user/hive/warehouse` identifies the NameNode location.
- `hdfs://nameservice1` specifies the new location and should match the value of the `dfs.nameservices` property.
- `tablePropKey` is a table property key whose value field may reference the HDFS NameNode location and hence may require an update. To update the Avro SerDe schema URL, specify `avro.schema.url` for this argument.
- `serdePropKey` is a SerDe property key whose value field may reference the HDFS NameNode location and hence may require an update. To update the Haivvero schema URL, specify `schema.url` for this argument.



Note: The `Hive metatool` is a best effort service that tries to update as many Hive metastore records as possible. If it encounters an error during the update of a record, it skips to the next record.

Configuring Hue to Work with HDFS HA

1. [Add the HttpFS](#) role.
2. After the command has completed, go to the **Hue** service.
3. Click the **Configuration** tab.
4. Locate the **HDFS Web Interface Role** property or search for it by typing its name in the Search box.
5. Select the **HttpFS** role you just created instead of the NameNode role, and save your changes.

- Restart the Hue service.

Configuring Impala to Work with HDFS HA

- Complete the steps to reconfigure the Hive metastore database, as described in the preceding section. Impala shares the same underlying database with Hive, to manage metadata for databases, tables, and so on.
- Issue the `INVALIDATE METADATA` statement from an Impala shell. This one-time operation makes all Impala daemons across the cluster aware of the latest settings for the Hive metastore database. Alternatively, restart the Impala service.

Configuring Oozie to Use HDFS HA

To configure an Oozie workflow to use HDFS HA, use the HDFS nameservice instead of the NameNode URI in the `<name-node>` element of the workflow.

Example:

```
<action name="mr-node">
  <map-reduce>
    <job-tracker>${jobTracker}</job-tracker>
    <name-node>hdfs://ha-nn
```

where `ha-nn` is the value of `dfs.nameservices` in `hdfs-site.xml`.

Administering an HDFS High Availability Cluster

Manually Failing Over to the Standby NameNode Using Cloudera Manager

If you are running a HDFS service with HA enabled, you can manually cause the active NameNode to failover to the standby NameNode. This is useful for planned downtime—for hardware changes, configuration changes, or software upgrades of your primary host.

- Go to the HDFS service.
- Click the **Instances** tab.
- Select **Actions > Manual Failover**. (This option does not appear if HA is not enabled for the cluster.)
- From the pop-up, select the NameNode that should be made active, then click **Manual Failover**.



Note: For advanced use only: You can set the **Force Failover** checkbox to force the selected NameNode to be active, irrespective of its state or the other NameNode's state. Forcing a failover will first attempt to failover the selected NameNode to active mode and the other NameNode to standby mode. It will do so even if the selected NameNode is in safe mode. If this fails, it will proceed to transition the selected NameNode to active mode. To avoid having two NameNodes be active, use this only if the other NameNode is either definitely stopped, or can be transitioned to standby mode by the first failover step.

- When all the steps have been completed, click **Finish**.

Cloudera Manager transitions the NameNode you selected to be the active NameNode, and the other NameNode to be the standby NameNode. HDFS should *never* have two active NameNodes.

Using the Command Line

To initiate a failover between two NameNodes, run the command `hdfs haadmin -failover`.

This command causes a failover from the first provided NameNode to the second. If the first NameNode is in the Standby state, this command simply transitions the second to the Active state without error. If the first NameNode is in the Active state, an attempt will be made to gracefully transition it to the Standby state. If this fails, the fencing methods (as configured by `dfs.ha.fencing.methods`) will be attempted in order until one of the methods succeeds. Only after this process will the second NameNode be transitioned to the Active state. If no fencing method succeeds, the second NameNode will not be transitioned to the Active state, and an error will be returned.



Note: Running `hdfs haadmin -failover` from the command line works whether you have configured HA from the command line or using Cloudera Manager. This means you can initiate a failover manually even if Cloudera Manager is unavailable.

Other `hdfs haadmin` Commands

After your HA NameNodes are configured and started, you will have access to some additional commands to administer your HA HDFS cluster. Specifically, you should familiarize yourself with the subcommands of the `hdfs haadmin` command.

This page describes high-level uses of some important subcommands. For specific usage information of each subcommand, you should run `hdfs haadmin -help <command>`.

`getServiceState`

`getServiceState` - determine whether the given NameNode is Active or Standby

Connect to the provided NameNode to determine its current state, printing either "standby" or "active" to `STDOUT` as appropriate. This subcommand might be used by `cron` jobs or monitoring scripts which need to behave differently based on whether the NameNode is currently Active or Standby.

`checkHealth`

`checkHealth` - check the health of the given NameNode

Connect to the provided NameNode to check its health. The NameNode is capable of performing some diagnostics on itself, including checking if internal services are running as expected. This command will return 0 if the NameNode is healthy, non-zero otherwise. One might use this command for monitoring purposes.



Note: The `checkHealth` command is not yet implemented, and at present will always return success, unless the given NameNode is completely down.

Using the `dfsadmin` command when HA is enabled

In previous versions of Hadoop, when HA was enabled, the `dfsadmin` command would not run operations on both active and standby NameNodes by default, even if the operations were permitted to run on both active and standby NameNodes. Due to an enhancement introduced in [HDFS-6507](#) (included in CDH 5.2), appropriate operations, such as `-refreshNodes`, `-refreshServiceAcl`, `-refreshUserToGroupsMappings`, and `-refreshSuperUserGroupsConfiguration`, now run on both active and standby NameNodes, unless you use the `-fs` option to specify a specific NameNode on which to run the operations.

Moving an HA NameNode to a New Host

Using the Command Line

Use the following steps to move one of the NameNodes to a new host.

In this example, the current NameNodes are called `nn1` and `nn2`, and the new NameNode is `nn2-alt`. The example assumes that `nn2-alt` is already a member of this CDH 5 HA cluster, that automatic failover is [configured](#) and that a JournalNode on `nn2` is to be moved to `nn2-alt`, in addition to NameNode service itself.

The procedure moves the NameNode and JournalNode services from `nn2` to `nn2-alt`, reconfigures `nn1` to recognize the new location of the JournalNode, and restarts `nn1` and `nn2-alt` in the new HA configuration.

Step 1: Make sure that `nn1` is the active NameNode

Make sure that the NameNode that is *not* going to be moved is active; in this example, `nn1` must be active. You can use the NameNodes' web UIs to see which is active; see [Start the NameNodes](#) on page 299.

If `nn1` is not the active NameNode, use the `hdfs haadmin -failover` command to initiate a failover from `nn2` to `nn1`:

```
hdfs haadmin -failover nn2 nn1
```

Step 2: Stop services on `nn2`

Once you've made sure that the node to be moved is inactive, stop services on that node: in this example, stop services on `nn2`. Stop the NameNode, the ZKFC daemon if this an automatic-failover deployment, and the JournalNode if you are moving it. Proceed as follows.

1. Stop the NameNode daemon:

```
$ sudo service hadoop-hdfs-namenode stop
```

2. Stop the ZKFC daemon if it is running:

```
$ sudo service hadoop-hdfs-zkfc stop
```

3. Stop the JournalNode daemon if it is running:

```
$ sudo service hadoop-hdfs-journalnode stop
```

4. Make sure these services are not set to restart on boot. If you are not planning to use `nn2` as a NameNode again, you may want remove the services.

Step 3: Install the NameNode daemon on `nn2-alt`

See the instructions for installing `hadoop-hdfs-namenode` in the *CDH 5 Installation Guide* under [Step 3: Install CDH 5 with YARN](#) or [Step 4: Install CDH 5 with MRv1](#).

Step 4: Configure HA on `nn2-alt`

See [Enabling HDFS HA](#) on page 288 for the properties to configure on `nn2-alt` in `core-site.xml` and `hdfs-site.xml`, and explanations and instructions. You should copy the values that are already set in the corresponding files on `nn2`.

- If you are relocating a JournalNode to `nn2-alt`, follow [these directions](#) to install it, but do not start it yet.
- If you are using automatic failover, make sure you follow the [instructions](#) for configuring the necessary properties on `nn2-alt` and initializing the HA state in Zookeeper.



Note: You do not need to shut down the cluster to do this if automatic failover is already configured as your failover method; shutdown is required only if you are switching from manual to automatic failover.

Step 5: Copy the contents of the `dfs.name.dir` and `dfs.journalnode.edits.dir` directories to `nn2-alt`

Use `rsync` or a similar tool to copy the contents of the `dfs.name.dir` directory, and the `dfs.journalnode.edits.dir` directory if you are moving the JournalNode, from `nn2` to `nn2-alt`.

Step 6: If you are moving a JournalNode, update `dfs.namenode.shared.edits.dir` on `nn1`

If you are relocating a JournalNode from `nn2` to `nn2-alt`, update `dfs.namenode.shared.edits.dir` in `hdfs-site.xml` on `nn1` to reflect the new hostname. See [this section](#) for more information about `dfs.namenode.shared.edits.dir`.

Step 7: If you are using automatic failover, install the `zkfc` daemon on `nn2-alt`

For instructions, see [Deploy Automatic Failover \(if it is configured\)](#) on page 300, but do not start the daemon yet.

Step 8: Start services on nn2-alt

Start the NameNode; start the ZKFC for automatic failover; and install and start a JournalNode if you want one to run on nn2-alt. Proceed as follows.

1. Start the JournalNode daemon:

```
$ sudo service hadoop-hdfs-journalnode start
```

2. Start the NameNode daemon:

```
$ sudo service hadoop-hdfs-namenode start
```

3. Start the ZKFC daemon:

```
$ sudo service hadoop-hdfs-zkfc start
```

4. Set these services to restart on boot; for example on a RHEL-compatible system:

```
$ sudo chkconfig hadoop-hdfs-namenode on  
$ sudo chkconfig hadoop-hdfs-zkfc on  
$ sudo chkconfig hadoop-hdfs-journalnode on
```

Step 9: If you are relocating a JournalNode, fail over to nn2-alt

```
hdfs haadmin -failover nn1 nn2-alt
```

Step 10: If you are relocating a JournalNode, restart nn1

Restart the NameNode daemon on nn1 to force it to re-read the configuration:

```
$ sudo service hadoop-hdfs-namenode stop  
$ sudo service hadoop-hdfs-namenode start
```

Converting From an NFS-mounted Shared Edits Directory to Quorum-based Storage Using Cloudera Manager

Converting a HA configuration from using an NFS-mounted shared edits directory to Quorum-based storage involves disabling the current HA configuration then enabling HA using Quorum-based storage.

1. [Disable HA](#).
2. Although the standby NameNode role is removed, its name directories are not deleted. Empty these directories.
3. [Enable HA with Quorum-based storage](#).

Using the Command Line

To switch from shared storage using NFS to Quorum-based storage, proceed as follows:

1. [Disable HA](#).
2. [Redeploy HA using Quorum-based storage](#).

Changing a Nameservice Name for Highly Available HDFS Using Cloudera Manager

For background on HDFS namespaces and HDFS high availability, see [Managing Federated Nameservices](#) on page 125 and [Enabling HDFS HA Using Cloudera Manager](#) on page 288.

1. Stop all services except ZooKeeper.
2. On a ZooKeeper server host, run `zookeeper-client`.

- a. Execute the following to remove the configured nameservice. This example assumes the name of the nameservice is **nameservice1**. You can identify the nameservice from the **Federation and High Availability** section on the **HDFS Instances** tab:

```
rmr /hadoop-ha/nameservice1
```

3. In the Cloudera Manager Admin Console, update the NameNode nameservice name.
 - a. Go to the HDFS service.
 - b. Click the **Configuration** tab.
 - c. Type `nameservice` in the Search field.
 - d. For the **NameNode Nameservice** property, type the nameservice name in the NameNode (*instance_name*) field. The name must be unique and can contain only alphanumeric characters.
 - e. Type `quorum` in the Search field.
 - f. For the **Quorum-based Storage Journal name** property, type the nameservice name in the NameNode (*instance_name*) field.
 - g. Click **Save Changes** to commit the changes.
4. Click the **Instances** tab.
5. In the Federation and High Availability pane, select **Actions > Initialize High Availability State in ZooKeeper**.
6. Go to the Hive service.
7. Select **Actions > Update Hive Metastore NameNodes**.
8. Go to the HDFS service.
9. Click the **Instances** tab.
10. Select the checkboxes next to the JournalNode role instances.
11. Select **Actions for Selected > Start**.
12. Click a **NameNode** role instance.
13. Select **Actions > Initialize Shared Edits Directory**.
14. Click the **Home** tab.
15. Redeploy client configuration files.
16. Start all services except ZooKeeper.

MapReduce (MRv1) and YARN (MRv2) High Availability

This section covers:

YARN (MRv2) ResourceManager High Availability

The YARN ResourceManager (RM) is responsible for tracking the resources in a cluster and scheduling applications (for example, MapReduce jobs). Before CDH 5, the RM was a single point of failure in a YARN cluster. The RM high availability (HA) feature adds redundancy in the form of an Active/Standby RM pair to remove this single point of failure. Furthermore, upon failover from the Standby RM to the Active, the applications can resume from their last check-pointed state; for example, completed map tasks in a MapReduce job are not re-run on a subsequent attempt. This allows events such the following to be handled without any significant performance effect on running applications.:

- Unplanned events such as machine crashes
- Planned maintenance events such as software or hardware upgrades on the machine running the ResourceManager.

RM HA requires ZooKeeper and HDFS services to be running.

Architecture

RM HA is implemented by means of an active-standby pair of RMs. On start-up, each RM is in the standby state: the process is started, but the state is not loaded. When transitioning to active, the RM loads the internal state from the designated state store and starts all the internal services. The stimulus to transition-to-active comes from either the

administrator (through the [CLI](#)) or through the integrated failover controller when [automatic failover](#) is enabled. The subsections that follow provide more details about the components of RM HA.

RM Restart

RM restart allows restarting the RM, while recovering the in-flight applications if recovery is enabled. To achieve this, the RM stores its internal state, primarily application-related data and tokens, to the `RMStateStore`; the cluster resources are re-constructed when the NodeManagers connect. The available alternatives for the state store are `MemoryRMStateStore` (a memory-based implementation), `FileSystemRMStateStore` (file system-based implementation; HDFS can be used for the file system), and `ZKRMStateStore` (ZooKeeper-based implementation).

Fencing

When running two RMs, a split-brain situation can arise where both RMs assume they are Active. To avoid this, only a single RM should be able to perform active operations and the other RM should be "fenced". The ZooKeeper-based state store (`ZKRMStateStore`) allows a single RM to make changes to the stored state, implicitly fencing the other RM. This is accomplished by the RM claiming exclusive create-delete permissions on the root znode. The ACLs on the root znode are automatically created based on the ACLs configured for the store; in case of secure clusters, Cloudera recommends that you set ACLs for the root node such that both RMs share read-write-admin access, but have exclusive create-delete access. The fencing is implicit and doesn't require explicit configuration (as fencing in HDFS and MRv1 does). You can plug in a custom "Fencer" if you choose to – for example, to use a different implementation of the state store.

Configuration and FailoverProxy

In an HA setting, you should configure two RMs to use different ports (for example, ports on different hosts). To facilitate this, YARN uses the notion of an RM Identifier (`rm-id`). Each RM has a unique `rm-id`, and all the RPC configurations (`<rpc-address>`; for example `yarn.resourcemanager.address`) for that RM can be configured via `<rpc-address>.<rm-id>`. Clients, ApplicationMasters, and NodeManagers use these RPC addresses to talk to the active RM automatically, even after a failover. To achieve this, they cycle through the list of RMs in the configuration. This is done automatically and doesn't require any configuration (as it does in HDFS and MapReduce (MRv1)).

Automatic Failover

By default, RM HA uses ZKFC (ZooKeeper-based failover controller) for automatic failover in case the active RM is unreachable or goes down. Internally, the **ActiveStandbyElector** is used to elect the Active RM. The failover controller runs as part of the RM (not as a separate process as in HDFS and MapReduce v1) and requires no further setup after the appropriate properties are [configured](#) in `yarn-site.xml`.

You can plug in a custom failover controller if you prefer.

Manual Transitions and Failover

You can use the [command-line tool](#) `yarn rmadmin` to transition a particular RM to active or standby state, to fail over from one RM to the other, to get the HA state of an RM, and to monitor an RM's health.

Configuring YARN (MRv2) ResourceManager High Availability Using Cloudera Manager

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

You can use Cloudera Manager to configure CDH 5 or later for ResourceManager high availability (HA). Cloudera Manager supports automatic failover of the ResourceManager. It does not provide a mechanism to manually force a failover through the Cloudera Manager user interface.



Important: Enabling or disabling HA will cause the previous monitoring history to become unavailable.

Enabling High Availability

1. Go to the YARN service.
2. Select **Actions > Enable High Availability**. A screen showing the hosts that are eligible to run a standby ResourceManager displays. The host where the current ResourceManager is running is not available as a choice.

3. Select the host where you want the standby ResourceManager to be installed, and click **Continue**. Cloudera Manager proceeds to execute a set of commands that stop the YARN service, add a standby ResourceManager, initialize the ResourceManager high availability state in ZooKeeper, restart YARN, and redeploy the relevant client configurations.
4. Work preserving recovery is enabled for the RM by default when you enable RM HA in Cloudera Manager. For more information, including instructions on disabling work preserving recovery, see [Work Preserving Recovery for YARN Components](#) on page 316.



Note: ResourceManager HA doesn't affect the JobHistory Server (JHS). JHS doesn't maintain any state, so if the host fails you can simply assign it to a new host. You can also enable process auto-restart by doing the following:

1. Go to the YARN service.
2. Click the **Configuration** tab.
3. Select **Scope > JobHistory Server**.
4. Select **Category > Advanced**.
5. Locate the **Automatically Restart Process** property or search for it by typing its name in the Search box.
6. Click **Edit Individual Values**
7. Select the JobHistory Server Default Group.
8. Restart the JobHistory Server role.

Disabling High Availability

1. Go to the YARN service.
2. Select **Actions > Disable High Availability**. A screen showing the hosts running the ResourceManagers displays.
3. Select which ResourceManager (host) you want to remain as the single ResourceManager, and click **Continue**. Cloudera Manager executes a set of commands that stop the YARN service, remove the standby ResourceManager and the Failover Controller, restart the YARN service, and redeploy client configurations.

Configuring YARN (MRv2) ResourceManager High Availability Using the Command Line

To configure and start ResourceManager HA, proceed as follows.

Stop the YARN daemons

Stop the MapReduce JobHistory service, ResourceManager service, and NodeManager on all nodes where they are running, as follows:

```
$ sudo service hadoop-mapreduce-historyserver stop
$ sudo service hadoop-yarn-resourcemanager stop
$ sudo service hadoop-yarn-nodemanager stop
```

Configure Manual Failover, and Optionally Automatic Failover

To configure failover:



Note:

Configure the following properties in `yarn-site.xml` as shown, whether you are configuring manual or automatic failover. They are sufficient to configure manual failover. You need to configure additional properties for automatic failover.

Name	Used On	Default Value	Recommended Value	Description
<code>yarn.resourcemanager.ha.enabled</code>	ResourceManager, NodeManager, Client	false	true	Enable HA

Name	Used On	Default Value	Recommended Value	Description
yarn.resourcemanager.ha.rm-ids	ResourceManager, NodeManager, Client	(None)	Cluster-specific, e.g., rm1,rm2	Comma-separated list of ResourceManager ids in this cluster.
yarn.resourcemanager.ha.id	ResourceManager	(None)	RM-specific, e.g., rm1	Id of the current ResourceManager. Must be set explicitly on each ResourceManager to the appropriate value.
yarn.resourcemanager.address.<rm-id>	ResourceManager, Client	(None)	Cluster-specific	The value of yarn.resourcemanager.address (Client-RM RPC) for this RM. Must be set for all RMs.
yarn.resourcemanager.scheduler.address.<rm-id>	ResourceManager, Client	(None)	Cluster-specific	The value of yarn.resourcemanager.scheduler.address (AM-RM RPC) for this RM. Must be set for all RMs.
yarn.resourcemanager.admin.address.<rm-id>	ResourceManager, Client/Admin	(None)	Cluster-specific	The value of yarn.resourcemanager.admin.address (RM administration) for this RM. Must be set for all RMs.
yarn.resourcemanager.resource-tracker.address.<rm-id>	ResourceManager, NodeManager	(None)	Cluster-specific	The value of yarn.resourcemanager.resource-tracker.address (NM-RM RPC) for this RM. Must be set for all RMs.
yarn.resourcemanager.webapp.address.<rm-id>	ResourceManager, Client	(None)	Cluster-specific	The value of yarn.resourcemanager.webapp.address (RM webapp) for this RM. Must be set for all RMs.
yarn.resourcemanager.recovery.enabled	ResourceManager	false	true	Enable job recovery on RM restart or failover.
yarn.resourcemanager.store.class	ResourceManager	org.apache.hadoop.yarn.server.resourcemanager.recovery.FileSystemRMStateStore	org.apache.hadoop.yarn.server.resourcemanager.recovery.ZooKeeperRMStateStore	The ResourceManageStateStore implementation to use to store the ResourceManager's internal state. The ZooKeeper-based store supports fencing

Name	Used On	Default Value	Recommended Value	Description
				implicitly. That it, it allows a single ResourceManager to make multiple changes at a time, and hence is recommended.
<code>yarn.resourcemanager.zk-address</code>	ResourceManager	(None)	Cluster-specific	The ZooKeeper quorum to use to store the ResourceManager's internal state.
<code>yarn.resourcemanager.zk-acl</code>	ResourceManager	<code>world:anyone:rwcd</code>	Cluster-specific	The ACLs the ResourceManager uses for the znode structure to store the internal state.
<code>yarn.resourcemanager.zk-state-store.root-node.acl</code>	ResourceManager	(None)	Cluster-specific	The ACLs used for the root node of the ZooKeeper state store. The ACLs set here should allow both ResourceManagers to read, write, and administer, with exclusive access to create and delete. If nothing is specified, the root node ACLs are automatically generated on the basis of the ACLs specified through <code>yarn.resourcemanager.zk-acl</code> . But that leaves a security hole in a secure setup.

To configure automatic failover:

Configure the following additional properties in `yarn-site.xml` to configure automatic failover.

Configure work preserving recovery:

Optionally, you can configure work preserving recovery for the Resource Manager and Node Managers. See [Work Preserving Recovery for YARN Components](#) on page 316.

Name	Used On	Default Value	Recommended Value	Description
<code>yarn.resourcemanager.ha.automatically-enabled</code>	ResourceManager	<code>true</code>	<code>true</code>	Enable automatic failover
<code>yarn.resourcemanager.ha.automatically-enabled</code>	ResourceManager	<code>true</code>	<code>true</code>	Use the EmbeddedElectorService

Name	Used On	Default Value	Recommended Value	Description
				to pick an Active RM from the ensemble
yarn.resourcemanager.cluster-id	ResourceManager	No default value.	Cluster-specific	Cluster name used by the ActiveStandbyElector to elect one of the ResourceManagers as leader.

The following is a sample `yarn-site.xml` showing these properties configured, including work preserving recovery for both RM and NM:

```
<configuration>
<!-- Resource Manager Configs -->
  <property>
    <name>yarn.resourcemanager.connect.retry-interval.ms</name>
    <value>2000</value>
  </property>
  <property>
    <name>yarn.resourcemanager.ha.enabled</name>
    <value>>true</value>
  </property>
  <property>
    <name>yarn.resourcemanager.ha.automatic-failover.enabled</name>
    <value>>true</value>
  </property>
  <property>
    <name>yarn.resourcemanager.ha.automatic-failover.embedded</name>
    <value>>true</value>
  </property>
  <property>
    <name>yarn.resourcemanager.cluster-id</name>
    <value>pseudo-yarn-rm-cluster</value>
  </property>
  <property>
    <name>yarn.resourcemanager.ha.rm-ids</name>
    <value>rm1,rm2</value>
  </property>
  <property>
    <name>yarn.resourcemanager.ha.id</name>
    <value>rm1</value>
  </property>
  <property>
    <name>yarn.resourcemanager.scheduler.class</name>
<value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.fair.FairScheduler</value>
  </property>
  <property>
    <name>yarn.resourcemanager.recovery.enabled</name>
    <value>>true</value>
  </property>
  <property>
    <name>yarn.resourcemanager.store.class</name>
    <value>org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKRMStateStore</value>
  </property>
  <property>
    <name>yarn.resourcemanager.zk-address</name>
    <value>localhost:2181</value>
  </property>
  <property>
    <name>yarn.app.mapreduce.am.scheduler.connection.wait.interval-ms</name>
    <value>5000</value>
  </property>
</property>
</configuration>
```

```

    <name>yarn.resourcemanager.work-preserving-recovery.enabled</name>
    <value>>true</value>
  </property>
<!-- RM1 configs -->
<property>
  <name>yarn.resourcemanager.address.rm1</name>
  <value>host1:23140</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address.rm1</name>
  <value>host1:23130</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.https.address.rm1</name>
  <value>host1:23189</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm1</name>
  <value>host1:23188</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address.rm1</name>
  <value>host1:23125</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address.rm1</name>
  <value>host1:23141</value>
</property>

<!-- RM2 configs -->
<property>
  <name>yarn.resourcemanager.address.rm2</name>
  <value>host2:23140</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address.rm2</name>
  <value>host2:23130</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.https.address.rm2</name>
  <value>host2:23189</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address.rm2</name>
  <value>host2:23188</value>
</property>
<property>
  <name>yarn.resourcemanager.resource-tracker.address.rm2</name>
  <value>host2:23125</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address.rm2</name>
  <value>host2:23141</value>
</property>

<!-- Node Manager Configs -->
<property>
  <description>Address where the localizer IPC is.</description>
  <name>yarn.nodemanager.localizer.address</name>
  <value>0.0.0.0:23344</value>
</property>
<property>
  <description>NM Webapp address.</description>
  <name>yarn.nodemanager.webapp.address</name>
  <value>0.0.0.0:23999</value>
</property>
<property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
</property>
<property>
  <name>yarn.nodemanager.local-dirs</name>

```

```
<value>/tmp/pseudo-dist/yarn/local</value>
</property>
<property>
  <name>yarn.nodemanager.log-dirs</name>
  <value>/tmp/pseudo-dist/yarn/log</value>
</property>
<property>
  <name>mapreduce.shuffle.port</name>
  <value>23080</value>
</property>
<property>
  <name>yarn.resourcemanager.work-preserving-recovery.enabled</name>
  <value>>true</value>
</property>
</configuration>
```

Re-start the YARN daemons

Start the MapReduce JobHistory server, ResourceManager, and NodeManager on all nodes where they were previously running, as follows:

```
$ sudo service hadoop-mapreduce-historyserver start
$ sudo service hadoop-yarn-resourcemanager start
$ sudo service hadoop-yarn-nodemanager start
```

Using `yarn radmin` to Administer ResourceManager HA

You can use `yarn radmin` on the command line to manage your ResourceManager HA deployment. `yarn radmin` has the following options related to RM HA:

```
[-transitionToActive serviceId]
[-transitionToStandby serviceId]
[-getServiceState serviceId]
[-checkHealth <serviceId>]
[-help <command>]
```

where *serviceId* is the `rm-id`.



Note: Even though `-help` lists the `-failover` option, it is not supported by `yarn radmin`.

Work Preserving Recovery for YARN Components

CDH 5.2 introduces *work preserving recovery* for the YARN ResourceManager and NodeManager. With work preserving recovery enabled, if a ResourceManager or NodeManager restarts, no in-flight work is lost. You can configure work preserving recovery separately for a ResourceManager or NodeManager.



Note: YARN does not support high availability for the Job History Server (JHS). If the JHS goes down, Cloudera Manager will restart it automatically.

Prerequisites

To use work preserving recovery for the ResourceManager, you need to first enable High Availability for the ResourceManager. See [YARN \(MRv2\) ResourceManager High Availability](#) on page 309 for more information.

Configuring Work Preserving Recovery Using Cloudera Manager

Use this procedure if you manage your CDH cluster using Cloudera Manager. Otherwise, see [Configuring Work Preserving Recovery Using the Command Line](#) on page 317.

If you use Cloudera Manager and you enable [YARN \(MRv2\) ResourceManager High Availability](#) on page 309, work preserving recovery is enabled by default for the ResourceManager. To disable the feature for the ResourceManager, change the value of `yarn.resourcemanager.work-preserving-recovery.enabled` to `false` in the `yarn-site.xml` using an advanced configuration snippet.

To enable the feature for a given NodeManager, edit the advanced configuration snippet for `yarn-site.xml` on that NodeManager, and set the value of `yarn.nodemanager.recovery.enabled` to `true`. For a given NodeManager, you can configure the directory on the local filesystem where state information is stored when work preserving recovery is enabled, by setting the value of `yarn.nodemanager.recovery.dir` to a local filesystem directory, using the same advanced configuration snippet. The default value is `${hadoop.tmp.dir}/yarn-nm-recovery`. This location usually points to the `/tmp` directory on the local filesystem. Because many operating systems do not preserve the contents of the `/tmp` directory across a reboot, Cloudera strongly recommends changing the location of `yarn.nodemanager.recovery.dir` to a different directory on the local filesystem. The [example](#) below uses `/home/cloudera/recovery`.

Configuring Work Preserving Recovery Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

After enabling ResourceManager High Availability, edit `yarn-site.xml` on the ResourceManager and all NodeManagers.

1. Set the value of `yarn.resourcemanager.work-preserving-recovery.enabled` to `true` to enable work preserving recovery for the ResourceManager, and set the value of `yarn.nodemanager.recovery.enabled` to `true` for the NodeManager.
2. For each NodeManager, configure the directory on the local filesystem where state information is stored when work preserving recovery is enabled, by setting the value of `yarn.nodemanager.recovery.dir` to a local filesystem directory. The default value is `${hadoop.tmp.dir}/yarn-nm-recovery`. This location usually points to the `/tmp` directory on the local filesystem. Because many operating systems do not preserve the contents of the `/tmp` directory across a reboot, Cloudera strongly recommends changing the location of `yarn.nodemanager.recovery.dir` to a different directory on the local filesystem. The [example](#) below uses `/home/cloudera/recovery`.
3. Configure a valid RPC address for the NodeManager, by setting `yarn.nodemanager.address` to an address with a specific port number (such as `0.0.0.0:45454`). Ephemeral ports (port 0, which is default) cannot be used for the NodeManager's RPC server, because this could cause the NodeManager to use different ports before and after a restart, preventing clients from connecting to the NodeManager after a restart. The NodeManager RPC address is also important for auxiliary services which run in a YARN cluster.

Auxiliary services should also be designed to support recoverability by reloading the previous state after a NodeManager restarts. An example auxiliary service, the ShuffleHandler service for MapReduce, follows the correct pattern for an auxiliary service which supports work preserving recovery of the NodeManager.

Example Configuration for Work Preserving Recovery

The following example configuration can be used with a Cloudera Manager advanced configuration snippet or added to `yarn-site.xml` directly if you do not use Cloudera Manager. Adjust the configuration to suit your environment.

```
<property>
  <name>yarn.resourcemanager.work-preserving-recovery.enabled</name>
  <value>>true</value>
  <description>Whether to enable work preserving recovery for the Resource
Manager</description>
</property>
<property>
  <name>yarn.nodemanager.recovery.enabled</name>
  <value>>true</value>
  <description>Whether to enable work preserving recovery for the Node
```

```
Manager</description>
</property>
<property>
  <name>yarn.nodemanager.recovery.dir</name>
  <value>/home/cloudera/recovery</value>
  <description>The location for stored state on the Node Manager, if work preserving
recovery
  is enabled.</description>
</property>
<property>
  <name>yarn.nodemanager.address</name>
  <value>0.0.0.0:45454</value>
</property>
```

MapReduce (MRv1) JobTracker High Availability

Follow the instructions in this section to configure high availability (HA) for JobTracker.

Configuring MapReduce (MRv1) JobTracker High Availability Using Cloudera Manager

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

You can use Cloudera Manager to configure CDH 4.3 or later for JobTracker high availability (HA). Although it is possible to configure JobTracker HA with CDH 4.2, it is not recommended. Rolling restart, decommissioning of TaskTrackers, and rolling upgrade of MapReduce from CDH 4.2 to CDH 4.3 are not supported when JobTracker HA is enabled.

Cloudera Manager supports automatic failover of the JobTracker. It does not provide a mechanism to manually force a failover through the Cloudera Manager user interface.



Important: Enabling or disabling JobTracker HA will cause the previous monitoring history to become unavailable.

Enabling JobTracker High Availability

The **Enable High Availability** workflow leads you through adding a second (standby) JobTracker:

1. Go to the MapReduce service.
2. Select **Actions > Enable High Availability**. A screen showing the hosts that are eligible to run a standby JobTracker displays. The host where the current JobTracker is running is not available as a choice.
3. Select the host where you want the Standby JobTracker to be installed, and click **Continue**.
4. Enter a directory location on the local filesystem for each JobTracker host. These directories will be used to store job configuration data.
 - You may enter more than one directory, though it is not required. The paths do not need to be the same on both JobTracker hosts.
 - If the directories you specify do not exist, they will be created with the appropriate permissions. If they already exist, they must be empty and have the appropriate permissions.
 - If the directories are not empty, Cloudera Manager will not delete the contents.
5. Optionally use the checkbox under Advanced Options to force initialize the ZooKeeper znode for auto-failover.
6. Click **Continue**. Cloudera Manager executes a set of commands that stop the MapReduce service, add a standby JobTracker and Failover controller, initialize the JobTracker high availability state in ZooKeeper, create the job status directory, restart MapReduce, and redeploy the relevant client configurations.

Disabling JobTracker High Availability

1. Go to the MapReduce service.
2. Select **Actions > Disable High Availability**. A screen showing the hosts running the JobTrackers displays.

3. Select which JobTracker (host) you want to remain as the single JobTracker, and click **Continue**. Cloudera Manager executes a set of commands that stop the MapReduce service, remove the standby JobTracker and the Failover Controller, restart the MapReduce service, and redeploy client configurations.

Configuring MapReduce (MRv1) JobTracker High Availability Using the Command Line

If you are running MRv1, you can configure the JobTracker to be highly available. You can configure either manual or automatic failover to a warm-standby JobTracker.



Note:

- As with [HDFS High Availability](#) on page 286, the JobTracker high availability feature is backward compatible; that is, if you do not want to enable JobTracker high availability, you can simply keep your existing configuration after updating your `hadoop-0.20-mapreduce`, `hadoop-0.20-mapreduce-jobtracker`, and `hadoop-0.20-mapreduce-tasktracker` packages, and start your services as before. You do not need to perform any of the actions described on this page.

To use the high availability feature, you must create a new configuration. This new configuration is designed such that all the hosts in the cluster can have the same configuration; you do not need to deploy different configuration files to different hosts depending on each host's role in the cluster.

In an HA setup, the `mapred.job.tracker` property is no longer a `host:port` string, but instead specifies a logical name to identify JobTracker instances in the cluster (active and standby). Each distinct JobTracker in the cluster has a different JobTracker ID. To support a single configuration file for all of the JobTrackers, the relevant configuration parameters are suffixed with the JobTracker logical name as well as the JobTracker ID.

The HA JobTracker is packaged separately from the original (non-HA) JobTracker.



Important: You cannot run both HA and non-HA JobTrackers in the same cluster. Do not install the HA JobTracker unless you need a highly available JobTracker. If you install the HA JobTracker and later decide to revert to the non-HA JobTracker, you will need to uninstall the HA JobTracker and re-install the non-HA JobTracker.

JobTracker HA reuses the `mapred.job.tracker` parameter in `mapred-site.xml` to identify a JobTracker active-standby pair. In addition, you must enable the existing `mapred.jobtracker.restart.recover`, `mapred.job.tracker.persist.jobstatus.active`, and `mapred.job.tracker.persist.jobstatus.hours` parameters, as well as a number of new parameters, as discussed below.

Use the sections that follow to install, configure and test JobTracker HA.

Replacing the non-HA JobTracker with the HA JobTracker

This section provides instructions for removing the non-HA JobTracker and installing the HA JobTracker.



Important: The HA JobTracker cannot be installed on a node on which the non-HA JobTracker is installed, and vice versa. If the JobTracker is installed, uninstall it following the instructions below before installing the HA JobTracker. Uninstall the non-HA JobTracker whether or not you intend to install the HA JobTracker on the same node.

Removing the non-HA JobTracker

You must remove the original (non-HA) JobTracker before you install and run the HA JobTracker. First, you need to stop the JobTracker and TaskTrackers.

To stop the JobTracker and TaskTrackers:

1. Stop the TaskTrackers: On each TaskTracker system:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker stop
```

2. Stop the JobTracker: On the JobTracker system:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker stop
```

3. Verify that the JobTracker and TaskTrackers have stopped:

```
$ ps -eaf | grep -i job  
$ ps -eaf | grep -i task
```

To remove the JobTracker:

- On Red Hat-compatible systems:

```
$ sudo yum remove hadoop-0.20-mapreduce-jobtracker
```

- On SLES systems:

```
$ sudo zypper remove hadoop-0.20-mapreduce-jobtracker
```

- On Ubuntu systems:

```
sudo apt-get remove hadoop-0.20-mapreduce-jobtracker
```

Installing the HA JobTracker

Use the following steps to install the HA JobTracker package, and optionally the ZooKeeper failover controller package (needed for automatic failover).

Step 1: Install the HA JobTracker package on two separate nodes

On each JobTracker node:

- On Red Hat-compatible systems:

```
$ sudo yum install hadoop-0.20-mapreduce-jobtrackerha
```

- On SLES systems:

```
$ sudo zypper install hadoop-0.20-mapreduce-jobtrackerha
```

- On Ubuntu systems:

```
sudo apt-get install hadoop-0.20-mapreduce-jobtrackerha
```

Step 2: (Optionally) install the failover controller package

If you intend to enable automatic failover, you need to install the failover controller package.



Note: The [instructions for automatic failover](#) assume that you have set up a ZooKeeper cluster running on three or more nodes, and have verified its correct operation by connecting using the ZooKeeper command-line interface (CLI). See the [ZooKeeper documentation](#) for instructions on how to set up a ZooKeeper ensemble.

Install the failover controller package as follows:

On each JobTracker node:

- On Red Hat-compatible systems:

```
$ sudo yum install hadoop-0.20-mapreduce-zkfc
```

- On SLES systems:

```
$ sudo zypper install hadoop-0.20-mapreduce-zkfc
```

- On Ubuntu systems:

```
sudo apt-get install hadoop-0.20-mapreduce-zkfc
```

Configuring and Deploying Manual Failover

Proceed as follows to configure manual failover:

1. [Configure the JobTrackers, TaskTrackers, and Clients](#)
2. [Start the JobTrackers](#)
3. [Activate a JobTracker](#)
4. [Verify that failover is working](#)

Step 1: Configure the JobTrackers, TaskTrackers, and Clients

Changes to existing configuration parameters

Property name	Default	Used on	Description
<code>mapred.job.tracker</code>	<code>local</code>	JobTracker, TaskTracker, client	In an HA setup, the logical name of the JobTracker active-standby pair. In a non-HA setup <code>mapred.job.tracker</code> is a <code>host:port</code> string specifying the JobTracker's RPC address, but in an HA configuration the logical name <i>must not</i> include a port number.
<code>mapred.jobtracker.restart.recover</code>	<code>false</code>	JobTracker	Whether to recover jobs that were running in the most recent active JobTracker. Must be set to <code>true</code> for JobTracker HA.
<code>mapred.job.tracker.persist.jobstatus.active</code>	<code>false</code>	JobTracker	Whether to make job status persistent in HDFS. Must be set to <code>true</code> for JobTracker HA.
<code>mapred.job.tracker.persist.jobstatus.hours</code>	<code>0</code>	JobTracker	The number of hours job status information is retained in HDFS. Must be greater than zero for JobTracker HA.

Property name	Default	Used on	Description
<code>mapred.job.tracker.persist.jobstatus.dir</code>	<code>/jobtracker/jobsInfo</code>	JobTracker	The HDFS directory in which job status information is kept persistently. The directory must exist and be owned by the <code>mapred</code> user.

New configuration parameters

Property name	Default	Used on	Description
<code>mapred.jobtrackers.<name></code>	None	JobTracker, TaskTracker, client	A comma-separated pair of IDs for the active and standby JobTrackers. The <code><name></code> is the value of <code>mapred.job.tracker</code> .
<code>mapred.jobtracker.rpc-address.<name>.<id></code>	None	JobTracker, TaskTracker, client	The RPC address of an individual JobTracker. <code><name></code> refers to the value of <code>mapred.job.tracker</code> ; <code><id></code> refers to one or other of the values in <code>mapred.jobtrackers.<name></code> .
<code>mapred.job.tracker.http.address.<name>.<id></code>	None	JobTracker, TaskTracker	The HTTP address of an individual JobTracker. (In a non-HA setup <code>mapred.job.tracker.http.address</code> (with no suffix) is the JobTracker's HTTP address.)
<code>mapred.ha.jobtracker.rpc-address.<name>.<id></code>	None	JobTracker, failover controller	The RPC address of the HA service protocol for the JobTracker. The JobTracker listens on a separate port for HA operations which is why this property exists in addition to <code>mapred.jobtracker.rpc-address.<name>.<id></code> .
<code>mapred.ha.jobtracker.http-redirect-address.<name>.<id></code>	None	JobTracker	The HTTP address of an individual JobTracker that should be used for HTTP redirects. The standby JobTracker will redirect all web traffic to the active, and will use this property to discover the URL to use for redirects. A property separate from <code>mapred.job.tracker.http.address.<name>.<id></code> is needed since the latter may be a wildcard bind address, such as <code>0.0.0.0:50030</code> , which is

Property name	Default	Used on	Description
			not suitable for making requests. Note also that <code>mapred.jobtracker.rpc-address.<name>.<id></code> is the HTTP redirect address for the JobTracker with ID <code><id></code> for the pair with the logical name <code><name></code> - that is, the address that should be used when that JobTracker is active, and <i>not</i> the address that should be redirected to when that JobTracker is the standby.
<code>mapred.ha.jobtracker.id</code>	None	JobTracker	The identity of this JobTracker instance. Note that this is optional since each JobTracker can infer its ID from the matching address in one of the <code>mapred.jobtracker.rpc-address.<name>.<id></code> properties. It is provided for testing purposes.
<code>mapred.client.failover.proxy.provider.<name></code>	None	TaskTracker, client	The failover provider class. The only class available is <code>org.apache.hadoop.mapred.ConfiguredFailoverProxyProvider</code> .
<code>mapred.client.failover.max.attempts</code>	15	TaskTracker, client	The maximum number of times to try to fail over.
<code>mapred.client.failover.sleep.base.millis</code>	500	TaskTracker, client	The time to wait before the first failover.
<code>mapred.client.failover.sleep.max.millis</code>	1500	TaskTracker, client	The maximum amount of time to wait between failovers (for exponential backoff).
<code>mapred.client.failover.connection.retries</code>	0	TaskTracker, client	The maximum number of times to retry between failovers.
<code>mapred.client.failover.connection.retries.on.timeouts</code>	0	TaskTracker, client	The maximum number of times to retry on timeouts between failovers.
<code>mapred.ha.fencing.methods</code>	None	failover controller	A list of scripts or Java classes that will be used to fence the active JobTracker during failover. Only one JobTracker should be active at any given time, but you can simply configure <code>mapred.ha.fencing.methods</code> as <code>shell(/bin/true)</code> since the JobTrackers fence themselves, and split-brain is

Property name	Default	Used on	Description
			avoided by the old active JobTracker shutting itself down if another JobTracker takes over.

Make changes and additions similar to the following to `mapred-site.xml` on each node.



Note: It is simplest to configure all the parameters on all nodes, even though not all of the parameters will be used on any given node. This also makes for robustness if you later change the roles of the nodes in your cluster.

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>

<!-- Put site-specific property overrides in this file. -->

<configuration>

  <property>
    <name>mapred.job.tracker</name>
    <value>logicaljt</value>
    <!-- host:port string is replaced with a logical name -->
  </property>

  <property>
    <name>mapred.jobtrackers.logicaljt</name>
    <value>jt1,jt2</value>
    <description>Comma-separated list of JobTracker IDs.</description>
  </property>

  <property>
    <name>mapred.jobtracker.rpc-address.logicaljt.jt1</name>
    <!-- RPC address for jt1 -->
    <value>myjt1.myco.com:8021</value>
  </property>

  <property>
    <name>mapred.jobtracker.rpc-address.logicaljt.jt2</name>
    <!-- RPC address for jt2 -->
    <value>myjt2.myco.com:8022</value>
  </property>

  <property>
    <name>mapred.job.tracker.http.address.logicaljt.jt1</name>
    <!-- HTTP bind address for jt1 -->
    <value>0.0.0.0:50030</value>
  </property>

  <property>
    <name>mapred.job.tracker.http.address.logicaljt.jt2</name>
    <!-- HTTP bind address for jt2 -->
    <value>0.0.0.0:50031</value>
  </property>

  <property>
    <name>mapred.ha.jobtracker.rpc-address.logicaljt.jt1</name>
    <!-- RPC address for jt1 HA daemon -->
    <value>myjt1.myco.com:8023</value>
  </property>

  <property>
    <name>mapred.ha.jobtracker.rpc-address.logicaljt.jt2</name>
    <!-- RPC address for jt2 HA daemon -->
    <value>myjt2.myco.com:8024</value>
  </property>
</configuration>
```

```
<property>
  <name>mapred.ha.jobtracker.http-redirect-address.logicaljt.jt1</name>
  <!-- HTTP redirect address for jt1 -->
  <value>myjt1.myco.com:50030</value>
</property>

<property>
  <name>mapred.ha.jobtracker.http-redirect-address.logicaljt.jt2</name>
  <!-- HTTP redirect address for jt2 -->
  <value>myjt2.myco.com:50031</value>
</property>

<property>
  <name>mapred.jobtracker.restart.recover</name>
  <value>true</value>
</property>

<property>
  <name>mapred.job.tracker.persist.jobstatus.active</name>
  <value>true</value>
</property>

<property>
  <name>mapred.job.tracker.persist.jobstatus.hours</name>
  <value>1</value>
</property>

<property>
  <name>mapred.job.tracker.persist.jobstatus.dir</name>
  <value>/jobtracker/jobsInfo</value>
</property>

<property>
  <name>mapred.client.failover.proxy.provider.logicaljt</name>
  <value>org.apache.hadoop.mapred.ConfiguredFailoverProxyProvider</value>
</property>

<property>
  <name>mapred.client.failover.max.attempts</name>
  <value>15</value>
</property>

<property>
  <name>mapred.client.failover.sleep.base.millis</name>
  <value>500</value>
</property>

<property>
  <name>mapred.client.failover.sleep.max.millis</name>
  <value>1500</value>
</property>

<property>
  <name>mapred.client.failover.connection.retries</name>
  <value>0</value>
</property>

<property>
  <name>mapred.client.failover.connection.retries.on.timeouts</name>
  <value>0</value>
</property>
<property>
  <name>mapred.ha.fencing.methods</name>
  <value>shell(/bin/true)</value>
</property>
</configuration>
```

**Note:**

In pseudo-distributed mode you need to specify `mapred.ha.jobtracker.id` for each JobTracker, so that the JobTracker knows its identity.

But in a fully-distributed setup, where the JobTrackers run on different nodes, there is no need to set `mapred.ha.jobtracker.id`, since the JobTracker can infer the ID from the matching address in one of the `mapred.jobtracker.rpc-address.<name>.<id>` properties.

Step 2: Start the JobTracker daemons

To start the daemons, run the following command on each JobTracker node:

```
$ sudo service hadoop-0.20-mapreduce-jobtrackerha start
```

Step 3: Activate a JobTracker**Note:**

- You must be the `mapred` user to use `mrhaadmin` commands.
- If Kerberos is enabled, do not use `sudo -u mapred` when using the `hadoop mrhaadmin` command. Instead, you must log in with the `mapred` Kerberos credentials (the short name must be `mapred`). See [Configuring Hadoop Security in CDH 5](#) for more information.

Unless automatic failover is configured, both JobTrackers will be in a standby state after the `jobtrackerha` daemons start up.

If Kerberos is not enabled, use the following commands:

To find out what state each JobTracker is in:

```
$ sudo -u mapred hadoop mrhaadmin -getServiceState <id>
```

where `<id>` is one of the values you [configured](#) in the `mapred.jobtrackers.<name>` property – `jt1` or `jt2` in our sample `mapred-site.xml` files.

To transition one of the JobTrackers to active and then verify that it is active:

```
$ sudo -u mapred hadoop mrhaadmin -transitionToActive <id>
$ sudo -u mapred hadoop mrhaadmin -getServiceState <id>
```

where `<id>` is one of the values you [configured](#) in the `mapred.jobtrackers.<name>` property – `jt1` or `jt2` in our sample `mapred-site.xml` files.

With Kerberos enabled, log in as the `mapred` user and use the following commands:

To log in as the `mapred` user and `kinit`:

```
$ sudo su - mapred
$ kinit -kt mapred.keytab mapred/<fully.qualified.domain.name>
```

To find out what state each JobTracker is in:

```
$ hadoop mrhaadmin -getServiceState <id>
```

where `<id>` is one of the values you [configured](#) in the `mapred.jobtrackers.<name>` property – `jt1` or `jt2` in our sample `mapred-site.xml` files.

To transition one of the JobTrackers to active and then verify that it is active:

```
$ hadoop mrhaadmin -transitionToActive <id>
$ hadoop mrhaadmin -getServiceState <id>
```

where <id> is one of the values you [configured](#) in the `mapred.jobtrackers.<name>` property – `jt1` or `jt2` in our sample `mapred-site.xml` files.

Step 4: Verify that failover is working

Use the following commands, depending whether or not Kerberos is enabled.

If Kerberos is not enabled, use the following commands:

To cause a failover from the currently active to the currently inactive JobTracker:

```
$ sudo -u mapred hadoop mrhaadmin -failover <id_of_active_JobTracker>
<id_of_inactive_JobTracker>
```

For example, if `jt1` is currently active:

```
$ sudo -u mapred hadoop mrhaadmin -failover jt1 jt2
```

To verify the failover:

```
$ sudo -u mapred hadoop mrhaadmin -getServiceState <id>
```

For example, if `jt2` should now be active:

```
$ sudo -u mapred hadoop mrhaadmin -getServiceState jt2
```

With Kerberos enabled, use the following commands:

To log in as the mapred user and kinit:

```
$ sudo su - mapred
$ kinit -kt mapred.keytab mapred/<fully.qualified.domain.name>
```

To cause a failover from the currently active to the currently inactive JobTracker:

```
$ hadoop mrhaadmin -failover <id_of_active_JobTracker> <id_of_inactive_JobTracker>
```

For example, if `jt1` is currently active:

```
$ hadoop mrhaadmin -failover jt1 jt2
```

To verify the failover:

```
$ hadoop mrhaadmin -getServiceState <id>
```

For example, if `jt2` should now be active:

```
$ hadoop mrhaadmin -getServiceState jt2
```

Configuring and Deploying Automatic Failover

To configure automatic failover, proceed as follows:

1. [Configure a ZooKeeper ensemble](#) (if necessary)
2. [Configure parameters for manual failover](#)
3. [Configure failover controller parameters](#)

4. [Initialize the HA state in ZooKeeper](#)
5. [Enable automatic failover](#)
6. [Verify automatic failover](#)

Step 1: Configure a ZooKeeper ensemble (if necessary)

To support automatic failover you need to set up a ZooKeeper ensemble running on three or more nodes, and verify its correct operation by connecting using the ZooKeeper command-line interface (CLI). See the [ZooKeeper documentation](#) for instructions on how to set up a ZooKeeper ensemble.



Note: If you are already using a ZooKeeper ensemble for [automatic failover](#), use the same ensemble for automatic JobTracker failover.

Step 2: Configure the parameters for manual failover

See the instructions for configuring the TaskTrackers and JobTrackers under [Configuring and Deploying Manual Failover](#).

Step 3: Configure failover controller parameters

Use the following additional parameters to configure a failover controller for each JobTracker. The failover controller daemons run on the JobTracker nodes.

New configuration parameters

Property name	Default	Configure on	Description
mapred.ha.automatic-failover.enabled	false	failover controller	Set to <code>true</code> to enable automatic failover.
mapred.ha.zkfc.port	8019	failover controller	The ZooKeeper failover controller port.
ha.zookeeper.quorum	None	failover controller	The ZooKeeper quorum (ensemble) to use for MRZKFailoverController.

Add the following configuration information to `mapred-site.xml`:

```
<property>
  <name>mapred.ha.automatic-failover.enabled</name>
  <value>true</value>
</property>

<property>
  <name>mapred.ha.zkfc.port</name>
  <value>8018</value>
  <!-- Pick a different port for each failover controller when running one machine -->
</property>
```

Add an entry similar to the following to `core-site.xml`:

```
<property>
  <name>ha.zookeeper.quorum</name>
  <value>zk1.example.com:2181,zk2.example.com:2181,zk3.example.com:2181 </value>
  <!-- ZK ensemble addresses -->
</property>
```




Note: If you have already [configured automatic failover for HDFS](#), this property is already properly configured; you use the same ZooKeeper ensemble for HDFS and JobTracker HA.

Step 4: Initialize the HA State in ZooKeeper

After you have configured the failover controllers, the next step is to initialize the required state in ZooKeeper. You can do so by running one of the following commands from one of the JobTracker nodes.



Note: The ZooKeeper ensemble must be running when you use this command; otherwise it will not work properly.

```
$ sudo service hadoop-0.20-mapreduce-zkfc init
```

or

```
$ sudo -u mapred hadoop mrzkfc -formatZK
```

This will create a znode in ZooKeeper in which the automatic failover system stores its data.



Note: If you are running a secure cluster, see also [Securing access to ZooKeeper](#).

Step 5: Enable automatic failover

To enable automatic failover once you have completed the configuration steps, you need only start the `jobtrackerha` and `zkfc` daemons.

To start the daemons, run the following commands on each JobTracker node:

```
$ sudo service hadoop-0.20-mapreduce-zkfc start
$ sudo service hadoop-0.20-mapreduce-jobtrackerha start
```

One of the JobTrackers will automatically transition to active.

Step 6: Verify automatic failover

After enabling automatic failover, you should test its operation. To do so, first locate the active JobTracker. To find out what state each JobTracker is in, use the following command:

```
$ sudo -u mapred hadoop mrhaadmin -getServiceState <id>
```

where `<id>` is one of the values you [configured](#) in the `mapred.jobtrackers.<name>` property – `jt1` or `jt2` in our sample `mapred-site.xml` files.



Note: You must be the `mapred` user to use `mrhaadmin` commands.

Once you have located your active JobTracker, you can cause a failure on that node. For example, you can use `kill -9 <pid of JobTracker>` to simulate a JVM crash. Or you can power-cycle the machine or its network interface to simulate different kinds of outages. After you trigger the outage you want to test, the other JobTracker should automatically become active within several seconds. The amount of time required to detect a failure and trigger a failover depends on the configuration of `ha.zookeeper.session-timeout.ms`, but defaults to 5 seconds.

If the test does not succeed, you may have a misconfiguration. Check the logs for the `zkfc` and `jobtrackerha` daemons to diagnose the problem.

Usage Notes

Using the JobTracker Web UI

To use the JobTracker Web UI, use the HTTP address of either JobTracker (that is, the value of `mapred.job.tracker.http.address.<name>.<id>` for either the active or the standby JobTracker). Note the following:

- If you use the URL of the standby JobTracker, you will be redirected to the active JobTracker.
- If you use the URL of a JobTracker that is down, you *will not* be redirected - you will simply get a "Not Found" error from your browser.

Turning off Job Recovery

After a failover, the newly active JobTracker by default restarts all jobs that were running when the failover occurred. For Sqoop 1 and HBase jobs, this is undesirable because they are not **idempotent** (that is, they do not behave the same way on repeated execution). For these jobs you should consider setting `mapred.job.restart.recover` to `false` in the job configuration (`JobConf`).

Cloudera Navigator Key Trustee Server High Availability



Warning: Key Trustee Server high availability applies to read operations only. If either Key Trustee Server fails, the KeyProvider automatically retries fetching keys from the functioning server. New write operations (for example, creating new encryption keys) are not allowed unless both Key Trustee Servers are operational.

Cloudera recommends monitoring both Key Trustee Servers. If a Key Trustee Server fails catastrophically, restore it from backup to a new host with the same hostname and IP address as the failed host. See [Backing Up and Restoring Key Trustee Server](#) for more information. Cloudera does not support PostgreSQL promotion to convert a passive Key Trustee Server to an active Key Trustee Server.

Configuring Key Trustee Server High Availability Using Cloudera Manager



Important: These instructions apply to using Cloudera Manager only. For package-based deployments, skip to the [Configuring Key Trustee Server High Availability Using the Command Line](#) on page 331 section.

For new installations, add the Key Trustee Server service to your cluster, following the instructions in [Adding a Service](#) on page 36. When customizing role assignments, assign the Active Key Trustee Server, Active Database, Passive Key Trustee Server, and Passive Database roles.



Important: If you are using SSH software other than OpenSSH, pre-create the SSH key before continuing:

```
$ sudo -u keytrustee ssh-keygen -t rsa -f /var/lib/keytrustee/.ssh/id_rsa
```

If you already have a Key Trustee Server service, and want to enable high availability, use the [Add Role Instances](#) wizard for the Key Trustee Server service instead to add the Passive Key Trustee Server and Passive Database roles.



Important: You *must* assign the Key Trustee Server and Database roles to the same host. Assign the Active Key Trustee Server and Active Database roles to one host, and the Passive Key Trustee Server and Passive Database roles to a separate host.

The remaining instructions apply to both new installations and adding a passive Key Trustee Server.

After completing the **Add a Service** or **Add Role Instances** wizard, the Passive Key Trustee Server and Passive Database roles fail to start. Complete the following manual actions to start these roles:

- Stop the Key Trustee Server service (**Key Trustee Server service** > **Actions** > **Stop**).
- Run the **Set Up Key Trustee Server Database** command (**Key Trustee Server service** > **Actions** > **Set Up Key Trustee Server Database**).
- Run the following command on the Active Key Trustee Server:

```
$ sudo rsync -zav --exclude .ssl /var/lib/keytrustee/.keytrustee
root@keytrustee02.example.com:/var/lib/keytrustee/.
```

Replace *keytrustee02.example.com* with the hostname of the Passive Key Trustee Server.

- Run the following command on the Passive Key Trustee Server:

```
$ sudo -u keytrustee /opt/cloudera/parcels/KEYTRUSTEE_SERVER/bin/ktadmin init
```

- Start the Key Trustee Server service (**Key Trustee Server service** > **Actions** > **Start**).



Important: Starting or restarting the Key Trustee Server service attempts to start the Active Database and Passive Database roles. If the Active Database is not running when the Passive Database attempts to start, the Passive Database fails to start. If this occurs, manually restart the Passive Database role after confirming that the Active Database role is running.

Enable Synchronous Replication

Key Trustee Server high availability requires synchronous replication to ensure that all rows in the database are inserted in at least two hosts, protecting against key loss.

To enable synchronous replication, run the following command on the active Key Trustee Server:

```
$ sudo -u keytrustee ktadmin enable-synchronous-replication --pg-rootdir
/var/lib/keytrustee/db
```

If you modified the default database location, replace */var/lib/keytrustee/db* with the modified path.

Configuring Key Trustee Server High Availability Using the Command Line

Install and configure a second Key Trustee Server following the instructions in [Installing Cloudera Navigator Key Trustee Server](#).

Once you have installed and configured the second Key Trustee Server, initialize the active Key Trustee Server by running the following commands on the active Key Trustee Server host:



Important: For Key Trustee Server 5.4.0 and higher, the `ktadmin init-master` command is deprecated, and should not be used. Use the `ktadmin init` command instead. If you are using SSH software other than OpenSSH, pre-create the SSH key on the active Key Trustee Server before continuing:

```
$ sudo -u keytrustee ssh-keygen -t rsa /var/lib/keytrustee/.ssh/id_rsa
```

```
$ sudo -u keytrustee ktadmin init --logdir /var/log/keytrustee --external-address
keytrustee01.example.com
$ sudo rsync -zav --exclude .ssl /var/lib/keytrustee/.keytrustee
root@keytrustee02.example.com:/var/lib/keytrustee/.
$ sudo -u keytrustee ktadmin db --bootstrap --port 11381 --pg-rootdir
/var/lib/keytrustee/db --slave keytrustee02.example.com
$ sudo /etc/init.d/keytrusteed start
```

Replace `keytrustee01.example.com` with the fully-qualified domain name (FQDN) of the active Key Trustee Server, `keytrustee02.example.com` with the FQDN of the passive Key Trustee Server, and `/var/lib/keytrustee/db` with the path to the directory you want to use to store the PostgreSQL database.

The `ktadmin init` command generates a self-signed certificate that the Key Trustee Server uses for HTTPS communication.

Initialize the passive Key Trustee Server by running the following commands on the passive host:

```
$ sudo -u keytrustee ktadmin init-slave --master keytrustee01.example.com --pg-rootdir
/var/lib/keytrustee/db --no-import-key --logdir /var/lib/keytrustee/.keytrustee/logs
--no-start
$ sudo -u keytrustee /usr/pgsql-9.3/bin/pg_ctl -D /var/lib/keytrustee/db start
$ sudo -u keytrustee ktadmin init --external-address keytrustee02.example.com
$ sudo /etc/init.d/keytrusteed start
```

Replace `keytrustee02.example.com` with the fully-qualified domain name (FQDN) of the passive Key Trustee Server, `keytrustee01.example.com` with the FQDN of the active Key Trustee Server, and `/var/lib/keytrustee/db` with the path to the directory you want to use to store the PostgreSQL database.

The `ktadmin init-slave` command performs an initial database sync by running the `pg_basebackup` command. The database directory must be empty for this step to work. For information on performing an incremental backup, see the [PostgreSQL documentation](#).



Note: The `/etc/init.d/postgresql` script does not work when the PostgreSQL database is started by Key Trustee Server, and cannot be used to monitor the status of the database. Use `ktadmin db [--start | --stop | --status]` instead.

The `ktadmin init` command generates a self-signed certificate that the Key Trustee Server uses for HTTPS communication. Instructions for using alternate certificates (for example, if you have obtained certificates from a trusted Certificate Authority) are provided later.

Enable Synchronous Replication

Key Trustee Server high availability requires synchronous replication to ensure that all rows in the database are inserted in at least two hosts, protecting against key loss.

To enable synchronous replication, run the following command on the active Key Trustee Server:

```
$ sudo -u keytrustee ktadmin enable-synchronous-replication --pg-rootdir
/var/lib/keytrustee/db
```

If you modified the default database location, replace `/var/lib/keytrustee/db` with the modified path.

(Optional) Replace Self-Signed Certificates with CA-Signed Certificates



Important: Key Trustee Server certificates must be issued to the fully-qualified domain name (FQDN) of the Key Trustee Server host. If you are using CA-signed certificates, ensure that the generated certificates use the FQDN, and not the short name.

If you have a CA-signed certificate for Key Trustee Server, see [Managing Key Trustee Server Certificates](#) for instructions on how to replace the self-signed certificates.

Recovering a Key Trustee Server

If a Key Trustee Server fails, restore it from backup as soon as possible. If the Key Trustee Server hosts fails completely, make sure that you restore the Key Trustee Server to a new host with the same hostname and IP address as the failed host.

For more information, see [Backing Up and Restoring Key Trustee Server](#).

Key Trustee KMS High Availability

CDH 5.4.0 and higher supports Key Trustee KMS high availability. If you have an existing standalone Key Trustee KMS service, use the following procedure to enable Key Trustee KMS high availability:

1. Back up the Key Trustee KMS private key and configuration directory. See [Backing Up Key Trustee Server Clients](#) for more information.
2. If you do not have a ZooKeeper service in your cluster, add one using the instructions in [Adding a Service](#) on page 36.
3. Run the [Add Role Instances](#) wizard for the Key Trustee KMS service (**Key Trustee KMS service** > **Actions** > **Add Role Instances**).
4. Click **Select hosts** and check the box for the host where you want to add the additional Key Management Server Proxy role. See [Resource Planning for Data at Rest Encryption](#) for considerations when selecting a host. Click **OK** and then **Continue**.
5. On the **Review Changes** page of the wizard, confirm the authorization code, organization name, and Key Trustee Server settings, and then click **Finish**.
6. Navigate to **Key Trustee KMS service** > **Configuration** and make sure that the **ZooKeeper Service** dependency is set to the ZooKeeper service for your cluster.
7. Restart the Key Trustee KMS service (**Key Trustee KMS service** > **Actions** > **Restart**).
8. Synchronize the Key Trustee KMS private key.



Warning: It is *very important* that you perform this step. Failure to do so leaves Key Trustee KMS in a state where keys are intermittently inaccessible, depending on which Key Trustee KMS host a client interacts with, because cryptographic key material encrypted by one Key Trustee KMS host cannot be decrypted by another. If you are already running multiple Key Trustee KMS hosts with different private keys, immediately [back up](#) all Key Trustee KMS hosts, and contact Cloudera Support for assistance correcting the issue.

To determine whether the Key Trustee KMS private keys are different, compare the MD5 hash of the private keys. On each Key Trustee KMS host, run the following command:

```
$ md5sum /var/lib/kms-keytrustee/keytrustee/.keytrustee/secring.gpg
```

If the outputs are different, contact Cloudera Support for assistance. Do not attempt to synchronize existing keys. If you overwrite the private key and do not have a backup, any keys encrypted by that private key are permanently inaccessible, and any data encrypted by those keys is permanently irretrievable. If you are configuring Key Trustee KMS high availability for the first time, continue synchronizing the private keys.

Cloudera recommends following security best practices and transferring the private key using offline media, such as a removable USB drive. For convenience (for example, in a development or testing environment where maximum security is not required), you can copy the private key over the network by running the following `rsync` command on the original Key Trustee KMS host:

```
rsync -zav /var/lib/kms-keytrustee/keytrustee/.keytrustee
root@ktkms02.example.com: /var/lib/kms-keytrustee/keytrustee/.
```

Replace `ktkms02.example.com` with the hostname of the Key Trustee KMS host that you are adding.

9. [Restart the cluster.](#)
10. Redeploy the client configuration (**Home** > **Cluster-wide** ▾ > **Deploy Client Configuration**).
11. Re-run the steps in [Validating Hadoop Key Operations](#).

High Availability for Other CDH Components

This section provides information on high availability for CDH components independently of HDFS. See also [Configuring Other CDH Components to Use HDFS HA](#) on page 302.

For details about HA for Impala, see [Using Impala through a Proxy for High Availability](#).

HBase High Availability

Most aspects of HBase are highly available in a standard configuration. A cluster typically consists of one Master and three or more RegionServers, with data stored in HDFS. To ensure that every component is highly available, configure one or more backup Masters. The backup Masters run on other hosts than the active Master.

Enabling HBase High Availability Using Cloudera Manager

1. Go to the HBase service.
2. Follow the process for [adding a role instance](#) and add a backup Master to a different host than the one on which the active Master is running.

Enabling HBase High Availability Using the Command Line

To configure backup Masters, create a new file in the `conf/` directory which will be distributed across your cluster, called `backup-masters`. For each backup Master you wish to start, add a new line with the hostname where the Master should be started. Each host that will run a Master needs to have all of the configuration files available. In general, it is a good practice to distribute the entire `conf/` directory across all cluster nodes.

After saving and distributing the file, restart your cluster for the changes to take effect. When the master starts the backup Masters, messages are logged. In addition, you can verify that an `HMaster` process is listed in the output of the `jps` command on the nodes where the backup Master should be running.

```
$ jps
15930 HRegionServer
16194 Jps
15838 HQuorumPeer
16010 HMaster
```

To stop a backup Master without stopping the entire cluster, first find its process ID using the `jps` command, then issue the `kill` command against its process ID.

```
$ kill 16010
```

HBase Read Replicas

CDH 5.4 introduces *read replicas*. Without read replicas, only one RegionServer services a read request from a client, regardless of whether RegionServers are colocated with other DataNodes that have local access to the same block.

This ensures consistency of the data being read. However, a RegionServer can become a bottleneck due to an underperforming RegionServer, network problems, or other reasons that could cause slow reads.

With read replicas enabled, the HMaster distributes read-only copies of regions (*replicas*) to different RegionServers in the cluster. One RegionServer services the default or *primary* replica, which is the only replica which can service write requests. If the RegionServer servicing the primary replica is down, writes will fail.

Other RegionServers serve the *secondaries* replicas, follow the primary RegionServer and only see committed updates. The secondary replicas are read-only, and are unable to service write requests. The secondary replicas can be kept up to date by reading the primary replica's HFiles at a set [interval](#) or by [replication](#). If they use the first approach, the secondary replicas may not have seen very recent updates to the data that the primary RegionServer has not yet flushed the memstore to HDFS. If the client receives the read response from a secondary replica, this is indicated by marking the read as "stale". Clients can detect whether or not the read result is stale and react accordingly.

Replicas are placed on different RegionServers, and on different racks when possible. This provides a measure of high availability (HA), as far as reads are concerned. If a RegionServer becomes unavailable, the regions it was serving can still be accessed by clients even before the region is taken over by a different RegionServer, using one of the secondary replicas. The reads may be stale until the entire WAL is processed by the new RegionServer for a given region.

For any given read request, a client can request a faster result even if it comes from a secondary replica, or if consistency is more important than speed, it can ensure that its request is serviced by the primary RegionServer. This allows you to decide the relative importance of consistency and availability, in terms of the [CAP Theorem](#), in the context of your application, or individual aspects of your application, using [Timeline Consistency](#) semantics.

Timeline Consistency

Timeline Consistency is a consistency model which allows for a more flexible standard of consistency than the default HBase model of *strong consistency*. A client can indicate the level of consistency it requires for a given read (Get or Scan) operation. The default consistency level is `STRONG`, meaning that the read request is only sent to the RegionServer servicing the region. This is the same behavior as when read replicas are not used. The other possibility, `TIMELINE`, sends the request to all RegionServers with replicas, including the primary. The client accepts the first response, which includes whether it came from the primary or a secondary RegionServer. If it came from a secondary, the client can choose to verify the read later or not to treat it as definitive.

Enabling Read Replica Support



Note:

Before you enable read-replica support, make sure to account for their increased heap memory requirements. Although no additional copies of HFile data are created, read-only replicas regions have the same memory footprint as normal regions and need to be considered when calculating the amount of increased heap memory required. For example, if your table requires 8 GB of heap memory, when you enable three replicas, you need about 24 GB of heap memory.

To enable support for read replicas in HBase, several properties must be set.

Table 12: HBase Read Replica Properties

Property Name	Default Value	Description
<code>hbase.region.replica.replication.enabled</code>	<code>false</code>	The mechanism for refreshing the secondary replicas. If set to <code>false</code> , secondary replicas are not guaranteed to be consistent at the row level. Secondary replicas are refreshed at intervals controlled by a timer (<code>hbase.regionserver.storefile.refresh.period</code>), and so are guaranteed to be at most that interval of milliseconds behind the primary RegionServer. Secondary replicas read from

Property Name	Default Value	Description
		<p>the HFile in HDFS, and have no access to writes that have not been flushed to the HFile by the primary RegionServer.</p> <p>If <code>true</code>, replicas are kept up to date using replication. and the column family has the attribute <code>REGION_MEMSTORE_REPLICATION</code> set to <code>false</code>, Using replication for read replication of <code>hbase:meta</code> is not supported, and <code>REGION_MEMSTORE_REPLICATION</code> must always be set to <code>false</code> on the column family.</p>
<code>hbase.regionserver.storefile.refresh.period</code>	0 (disabled)	<p>The period, in milliseconds, for refreshing the store files for the secondary replicas. The default value of 0 indicates that the feature is disabled. Secondary replicas update their store files from the primary RegionServer at this interval.</p> <p>If refreshes occur too often, this can create a burden for the NameNode. If refreshes occur too infrequently, secondary replicas will be less consistent with the primary RegionServer.</p>
<code>hbase.master.loadbalancer.class</code>	<p><code>org.apache.hadoop.hbase.master.loadbalancer.StochasticLoadBalancer</code></p> <p><code>balancer.StochasticBalancer</code></p> <p>(the class name is split for formatting purposes)</p>	<p>The Java class used for balancing the load of all HBase clients. The default implementation is the <code>StochasticLoadBalancer</code>, which is the only load balancer that supports reading data from secondary RegionServers.</p>
<code>hbase.ipc.client.allowsInterrupt</code>	<code>true</code>	<p>Whether or not to enable interruption of RPC threads at the client. The default value of <code>true</code> enables primary RegionServers to access data from other regions' secondary replicas.</p>
<code>hbase.client.primaryCallTimeout.get</code>	10 ms	<p>The timeout period, in milliseconds, an HBase client's will wait for a response before the read is submitted to a secondary replica if the read request allows timeline consistency. The default value is 10. Lower values increase the number of remote procedure calls while lowering latency.</p>
<code>hbase.client.primaryCallTimeout.multiget</code>	10 ms	<p>The timeout period, in milliseconds, before an HBase client's multi-get request, such as <code>HTable.get(List<GET>)</code>, is submitted to a secondary replica if the multi-get request allows timeline consistency. Lower values increase the number of remote procedure calls while lowering latency.</p>

Configure Read Replicas Using Cloudera Manager

1. Select **Clusters > HBase**.
2. Click the **Configuration** tab.
3. Select **Scope > HBase or HBase Service-Wide**.
4. Select **Category > Advanced**.
5. Locate the **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml** property or search for it by typing its name in the Search box.
6. Using the same XML syntax as [Configure Read Replicas Using the Command Line](#) on page 337 and the chart above, create a configuration and paste it into the text field.
7. Click **Save Changes** to commit the changes.

Configure Read Replicas Using the Command Line



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Add the properties from [Table 12: HBase Read Replica Properties](#) on page 335 to `hbase-site.xml` on each RegionServer in your cluster, and configure each of them to a value appropriate to your needs. The following example configuration shows the syntax.

```
<property>
  <name>hbase.regionserver.storefile.refresh.period</name>
  <value>0</value>
</property>
<property>
  <name>hbase.ipc.client.allowsInterrupt</name>
  <value>true</value>
  <description>Whether to enable interruption of RPC threads at the client. The default
  value of true is
  required to enable Primary RegionServers to access other RegionServers in secondary
  mode. </description>
</property>
<property>
  <name>hbase.client.primaryCallTimeout.get</name>
  <value>10</value>
</property>
<property>
  <name>hbase.client.primaryCallTimeout.multiget</name>
  <value>10</value>
</property>
```

2. Restart each RegionServer for the changes to take effect.

Activating Read Replicas On a Table

After enabling read replica support on your RegionServers, configure the tables for which you want read replicas to be created. Keep in mind that each replica increases the amount of storage used by HBase in HDFS.

At Table Creation

To create a new table with read replication capabilities enabled, set the `REGION_REPLICATION` property on the table. Use a command like the following, in HBase Shell:

```
hbase> create 'myTable', 'myCF', {REGION_REPLICATION => '3'}
```

By Altering an Existing Table

You can also alter an existing column family to enable or change the number of read replicas it propagates, using a command similar to the following. The change will take effect at the next major compaction.

```
hbase> disable 'myTable'  
hbase> alter 'myTable', 'myCF', {REGION_REPLICATION => '3'}  
hbase> enable 'myTable'
```

Requesting a Timeline-Consistent Read

To request a timeline-consistent read in your application, use the `get.setConsistency(Consistency.TIMELINE)` method before performing the `Get` or `Scan` operation.

To check whether the result is stale (comes from a secondary replica), use the `isStale()` method of the result object. Use the following examples for reference.

Get Request

```
Get get = new Get(key);  
get.setConsistency(Consistency.TIMELINE);  
Result result = table.get(get);
```

Scan Request

```
Scan scan = new Scan();  
scan.setConsistency(CONSISTENCY.TIMELINE);  
ResultScanner scanner = table.getScanner(scan);  
Result result = scanner.next();
```

Scan Request to a Specific Replica

This example overrides the normal behavior of sending the read request to all known replicas, and only sends it to the replica specified by ID.

```
Scan scan = new Scan();  
scan.setConsistency(CONSISTENCY.TIMELINE);  
scan.setReplicaId(2);  
ResultScanner scanner = table.getScanner(scan);  
Result result = scanner.next();
```

Detecting a Stale Result

```
Result result = table.get(get);  
if (result.isStale()) {  
    ...  
}
```

Getting and Scanning Using HBase Shell

You can also request timeline consistency using HBase Shell, allowing the result to come from a secondary replica.

```
hbase> get 'myTable', 'myRow', {CONSISTENCY => "TIMELINE"}  
hbase> scan 'myTable', {CONSISTENCY => 'TIMELINE'}
```

Hive Metastore High Availability

You can enable Hive metastore high availability (HA), so that your cluster is resilient to failures due to a metastore that becomes unavailable. Each metastore is independent; they do not use a quorum.

Prerequisites

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

Limitations


Sentry HDFS synchronization does not support Hive metastore HA.

Enabling Hive Metastore High Availability Using Cloudera Manager

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

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Select **Scope > Hive Metastore Server**.
4. Select **Category > Advanced**.
5. Locate the **Hive Metastore Delegation Token Store** property or search for it by typing its name in the Search box.
6. If you use a secure cluster, enable the Hive token store by setting the value of the **Hive Metastore Delegation Token Store** property to `org.apache.hadoop.hive.thrift.DBTokenStore`.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

7. Click **Save Changes** to commit the changes.
8. Click the **Instances** tab.
9. Click **Add Role Instances**.
10. Click the text field under **Hive Metastore Server**.
11. Select a host on which to run the additional metastore and click **OK**.
12. Click **Continue** and click **Finish**.
13. Check the checkbox next to the new **Hive Metastore Server** role.
14. Select **Actions for Selected > Start**, and click **Start** to confirm.
15. Click  to display the stale configurations page.
16. Click **Restart Cluster** and click **Restart Now**.
17. Click **Finish** after the cluster finishes restarting.

Enabling Hive Metastore High Availability Using the Command Line

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



Important:

- If you use Cloudera Manager, do not use these command-line instructions.
- This information applies specifically to CDH 5.4.x. If you use an earlier version of CDH, see the documentation for that version located at [Cloudera Documentation](#).

1. Configure Hive on each of the cluster hosts where you want to run a metastore, following the instructions at [Configuring the Hive Metastore](#).

2. On the server where the master metastore instance runs, edit the `/etc/hive/conf.server/hive-site.xml` file, setting the `hive.metastore.uris` property's value to a list of URIs where a Hive metastore is available for failover.

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

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

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

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

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

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

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

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

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

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

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

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

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

```
$ sudo service hive-metastore stop
$ /usr/lib/hive/bin/beeline
beeline> !connect jdbc:hive2://metastore2.example.com:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000> SHOW TABLES;
show tables;
+-----+
| tab_name |
+-----+
+-----+
No rows selected (0.238 seconds)
0: jdbc:hive2://localhost:10000>
```

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

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

Hue High Availability

To make the Hue service highly available, you configure at least two instances of the Hue service, each on a different host. You also configure the **nginx** load balancer to direct traffic to an alternate host if the primary host becomes

unavailable. This topic provides basic configuration information for using, installing, and configuring nginx. For information on advanced configurations, see the [nginx documentation](#).

Prerequisites

- Network access via SSH to the host machines with the Hue role.

Preparing a Cluster for Hue High Availability

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

Follow these steps to prepare your cluster for Hue high availability.

1. Configuring High Availability for a New Hue Service:

Add the Hue service to your cluster using the **Add a Service** wizard. See [Adding the Hue Service](#) on page 165. Hue high availability requires at least two instances of the Hue Server role in your cluster. Starting with CDH 5.4.1, you can add a new Hue service with multiple Hue Server roles.

For Kerberized clusters, the Add Service wizard will automatically add a colocated Kerberos Ticket Renewer role for each Hue Server role instance.

Configuring High Availability for an Existing Hue Service:

To add a Hue Server role to an existing Hue service on a Kerberized cluster, use the Add Role Instances wizard. Cloudera Manager will generate a validation error if the new Hue Server role does not have a colocated Kerberos Ticket Renewer role. To add a Hue role instance, see [Adding a Hue Role Instance](#) on page 165.

2. Configure the backend database for high availability. For more information, consult the vendor documentation for the database that you configured for Hue.
3. Configure the Hue instances in your cluster to connect to the backend database. See [Using an External Database for Hue Using Cloudera Manager](#) on page 168.

Installing the nginx Load Balancer

To enable high availability for Hue, install the nginx load balancer on one of the Hue instances. Clients access this instance through a designated port number, and the nginx load balancer directs the request to one of the Hue server instances.

1. Using SSH, log in as the **root** user to the host machine of one of the Hue instances.
2. Run the following command to install nginx:

Red Hat/Centos:

```
yum install nginx
```

Debian/Ubuntu:

```
apt-get install nginx
```

3. Create the following configuration file:

```
/etc/nginx/conf.d/hue.conf
```

4. Using the following text as a template, enter the configuration for your cluster in the `hue.conf` file:

```
server {
    server_name NGINX_HOSTNAME;
    charset utf-8;

    listen 8001;

    # Or if running hue on https://
```

```

## listen 8001 ssl;
## ssl_certificate /path/to/ssl/cert;
## ssl_certificate_key /path/to/ssl/key;

location / {
    proxy_pass http://hue;

    # Or if the upstream Hue instances are running behind https://
    ## proxy_pass https://hue;
}

location /static/ {
    # Uncomment to expose the static file directories.
    ## autoindex on;

    # If Hue was installed with packaging install:
    alias /usr/lib/hue/build/static/;

    # Or if on a parcel install:
    ## alias /opt/cloudera/parcels/CDH/lib/hue/build/static/;

    expires 30d;
    add_header Cache-Control public;
}
}

upstream hue {
    ip_hash;

    # List all the Hue instances here for high availability.
    server HUE_HOST1:8888 max_fails=3;
    server HUE_HOST2:8888 max_fails=3;
    ...
}

```

5. Update the following items in the `hue.conf` file:

- Replace `NGINX_HOSTNAME` with the URL of the host where you installed nginx. For example:

```
server_name myhost-2.myco.com;
```

- In the `location/static` block, comment or uncomment the `alias` lines depending on whether your cluster was installed using parcels or packages. (The comment indicator is `#`.)
- In the `upstream hue` block, list all the hostnames, including port number, of the Hue instances in your cluster. For example:

```
server myhost-1.myco.com:8888 max_fails=3;
server myhost-2.myco.com:8888 max_fails=3;
server myhost-4.myco.com:8888 max_fails=3;
```

- If the Hue service in your cluster is configured to use SSL:
 - Uncomment these lines, and replace the path names with the correct values for your cluster:

```
## listen 8001 ssl;
## ssl_certificate /path/to/ssl/cert;
## ssl_certificate_key /path/to/ssl/key;
```

- Uncomment the following line in the `location /` block:

```
## proxy_pass https://hue;
```

and comment out the following line:

```
proxy_pass http://hue;
```

- Comment out the following line:

```
listen 8001
```

6. Run the following command to start nginx:

```
sudo service nginx start
```

7. Test your installation by opening the Hue application in a web browser, using the following URL:

- Without SSL: `http://NGINX_HOSTNAME:8001`
- With SSL: `https://NGINX_HOSTNAME:8001`

Where `NGINX_HOSTNAME` is the name of the host machine where you installed nginx.



Note: You can use the following page to check if Hue is still alive:

`http://<HUE_HOST>/desktop/debug/is_alive`. The 200 status code is returned, which indicates that Hue is still running.

8. Test high availability:

- a. Stop the Hue service instance on the host where you installed nginx.
- b. Access the Hue application as described in the previous step. If you can connect to the Hue application, you have successfully enabled high availability.

Llama High Availability

Llama high availability (HA) uses an active-standby architecture, in which the active Llama is automatically elected using the ZooKeeper-based ActiveStandbyElector. The active Llama accepts RPC Thrift connections and communicates with YARN. The standby Llama monitors the leader information in ZooKeeper, but doesn't accept RPC Thrift connections.

Fencing

Only one of the Llamas should be active to ensure the resources are not partitioned. Llama uses ZooKeeper access control lists (ACLs) to claim exclusive ownership of the cluster when transitioning to active, and monitors this ownership periodically. If another Llama takes over, the first one realizes it within this period.

Reclaiming Cluster Resources

To claim resources from YARN, Llama spawns YARN applications and runs unmanaged ApplicationMasters. When a Llama goes down, the resources allocated to all the YARN applications spawned by it are not reclaimed until YARN times out those applications (the default timeout is 10 minutes). On Llama failure, these resources are reclaimed by means of a Llama that kills any YARN applications spawned by this pair of Llamas.

Enabling Llama High Availability Using Cloudera Manager

You can enable Llama high availability when you [enable integrated resource management](#). If you chose to create a single Llama instance at that time, follow these steps to enable Llama high availability:

1. Go to the Impala service.
2. Select **Actions > Enable High Availability**.
3. Click the **Impala Llama ApplicationMaster Hosts** field to display a dialog for choosing Llama hosts.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

4. Specify or select one or more hosts and click **OK**.
5. Click **Continue**.
6. Click **Continue**. A progress screen displays with a summary of the wizard actions.
7. Click **Finish**.

Disabling Llama High Availability Using Cloudera Manager

You can disable Llama high availability when you [disable integrated resource management](#). If you choose not to disable integrated resource management, follow these steps to disable Llama high availability:

1. Go to the Impala service.
2. Select **Actions > Disable High Availability**.
3. Choose the host on which Llama runs after high availability is disabled.
4. Click **Continue**. A progress screen displays with a summary of the wizard actions.
5. Click **Finish**.

Enabling Llama High Availability Using the Command Line

1. Configure Llama HA by modifying the following configuration properties in `/etc/llama/conf/llama-site.xml`. There is no need for any additional daemons.

Property	Description	Default	Recommended
<code>llama.am.cluster.id</code>	Cluster ID of the Llama pair, used to differentiate between different Llamas	<code>llama</code>	[cluster-specific]
<code>llama.am.ha.enabled*</code>	Whether to enable Llama HA	<code>false</code>	<code>true</code>
<code>llama.am.ha.zk-quorum*</code>	ZooKeeper quorum to use for leader election and fencing		[cluster-specific]
<code>llama.am.ha.zk-base</code>	Base znode for leader election and fencing data	<code>/llama</code>	[cluster-specific]
<code>llama.am.ha.zk-timeout-ms</code>	The session timeout, in milliseconds, for connections to ZooKeeper quorum	<code>10000</code>	<code>10000</code>
<code>llama.am.ha.zk-acl</code>	ACLs to control access to ZooKeeper	<code>world:anyone:rwcd</code>	[cluster-specific]
<code>llama.am.ha.zk-auth</code>	Authentication information to go with the ACLs		[cluster-acl-specific]

*Required configurations

2. Configure the [Impala daemon to use the HA Llama](#).

Configuring Oozie for High Availability

In CDH 5, you can configure multiple active Oozie servers against the same database. Oozie high availability is “active-active” or “hot-hot” so that both Oozie servers are active at the same time, with no failover. High availability for Oozie is supported in both MRv1 and MRv2 (YARN).

Requirements

The requirements for Oozie high availability are:

- Multiple active Oozie servers, preferably identically configured.
- JDBC JAR in the same location across all Oozie hosts (for example, `/var/lib/oozie/`).
- External database that supports multiple concurrent connections, preferably with HA support. The default Derby database does not support multiple concurrent connections.
- ZooKeeper ensemble with distributed locks to control database access, and service discovery for log aggregation.
- Load balancer (preferably with HA support, for example [HAProxy](#)), virtual IP, or round-robin DNS to provide a single entry point (of the multiple active servers), and for callbacks from the Application Master or JobTracker.

For information on setting up SSL communication with Oozie HA enabled, see [Additional Considerations when Configuring SSL for Oozie HA](#).

Configuring Oozie High Availability Using Cloudera Manager

Minimum Required Role: [Full Administrator](#)



Important: Enabling or disabling high availability makes the previous monitoring history unavailable.

Enabling Oozie High Availability

1. Ensure that the [requirements](#) are satisfied.
2. In the Cloudera Manager Admin Console, go to the Oozie service.
3. Select **Actions** > **Enable High Availability** to see eligible Oozie server hosts. The host running the current Oozie server is not eligible.
4. Select the host on which to install an additional Oozie server and click **Continue**.
5. Specify the host and port of the Oozie load balancer, and click **Continue**. Cloudera Manager stops the Oozie servers, adds another Oozie server, initializes the Oozie server High Availability state in ZooKeeper, configures Hue to reference the Oozie load balancer, and restarts the Oozie servers and dependent services.

Disabling Oozie High Availability

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Select **Actions** > **Disable High Availability** to see all hosts currently running Oozie servers.
3. Select the one host to run the Oozie server and click **Continue**. Cloudera Manager stops the Oozie service, removes the additional Oozie servers, configures Hue to reference the Oozie service, and restarts the Oozie service and dependent services.

Configuring Oozie High Availability Using the Command Line

For installation and configuration instructions for configuring Oozie HA using the command line, see <https://archive.cloudera.com/cdh5/cdh/5/oozie>.

Search High Availability

Mission critical, large-scale online production systems need to make progress without downtime despite some issues. Cloudera Search provides two routes to configurable, highly available, and fault-tolerant data ingestion:

- Near Real Time (NRT) ingestion using the Flume Solr Sink
- MapReduce based batch ingestion using the MapReduceIndexerTool

Production versus Test Mode

Some exceptions are generally transient, in which case the corresponding task can simply be retried. For example, network connection errors or timeouts are recoverable exceptions. Conversely, tasks associated with an unrecoverable exception cannot simply be retried. Corrupt or malformed parser input data, parser bugs, and errors related to unknown Solr schema fields produce unrecoverable exceptions.

Different modes determine how Cloudera Search responds to different types of exceptions.

- **Configuration parameter `isProductionMode=false`** (Non-production mode or test mode): Default configuration. Cloudera Search throws exceptions to quickly reveal failures, providing better debugging diagnostics to the user.
- **Configuration parameter `isProductionMode=true`** (Production mode): Cloudera Search logs and ignores unrecoverable exceptions, enabling mission-critical large-scale online production systems to make progress without downtime, despite some issues.



Note: Categorizing exceptions as recoverable or unrecoverable addresses most cases, though it is possible that an unrecoverable exception could be accidentally misclassified as recoverable. Cloudera provides the `isIgnoringRecoverableExceptions` configuration parameter to address such a case. In a production environment, if an unrecoverable exception is discovered that is classified as recoverable, change `isIgnoringRecoverableExceptions` to `true`. Doing so allows systems to make progress and avoid retrying an event forever. This configuration flag should only be enabled if a misclassification bug has been identified. Please report such bugs to Cloudera.

If Cloudera Search throws an exception according to the rules described above, the caller, meaning Flume Solr Sink and MapReduceIndexerTool, can catch the exception and retry the task if it meets the criteria for such retries.

Near Real Time Indexing with the Flume Solr Sink

The Flume Solr Sink uses the settings established by the `isProductionMode` and `isIgnoringRecoverableExceptions` parameters. If a SolrSink does nonetheless receive an exception, the SolrSink rolls the transaction back and pauses. This causes the Flume channel, which is essentially a queue, to redeliver the transaction's events to the SolrSink approximately five seconds later. This redelivering of the transaction event retries the ingest to Solr. This process of rolling back, backing off, and retrying continues until ingestion eventually succeeds.

Here is a corresponding example Flume configuration file `flume.conf`:

```
agent.sinks.solrSink.isProductionMode = true
agent.sinks.solrSink.isIgnoringRecoverableExceptions = true
```

In addition, Flume SolrSink automatically attempts to load balance and failover among the hosts of a SolrCloud before it considers the transaction rollback and retry. Load balancing and failover is done with the help of ZooKeeper, which itself can be configured to be highly available.

Further, Cloudera Manager can configure Flume so it automatically restarts if its process crashes.

To tolerate extended periods of Solr downtime, you can configure Flume to use a high-performance transactional persistent queue in the form of a [FileChannel](#). A FileChannel can use any number of local disk drives to buffer significant amounts of data. For example, you might buffer many terabytes of events corresponding to a week of data. Further, using the [Replicating Channel Selector](#) Flume feature, you can configure Flume to replicate the same data both into HDFS as well as into Solr. Doing so ensures that if the Flume SolrSink channel runs out of disk space, data delivery is still delivered to HDFS, and this data can later be ingested from HDFS into Solr using MapReduce.

Many machines with many Flume Solr Sinks and FileChannels can be used in a failover and load balancing configuration to improve high availability and scalability. Flume SolrSink servers can be either co-located with live Solr servers serving end user queries, or Flume SolrSink servers can be deployed on separate industry standard hardware for improved scalability and reliability. By spreading indexing load across a large number of Flume SolrSink servers you can improve

scalability. Indexing load can be replicated across multiple Flume SolrSink servers for high availability, for example using Flume features such as [Load balancing Sink Processor](#).

Batch Indexing with MapReduceIndexerTool

The Mappers and Reducers of the MapReduceIndexerTool follow the settings established by the `isProductionMode` and `isIgnoringRecoverableExceptions` parameters. However, if a Mapper or Reducer of the MapReduceIndexerTool does receive an exception, it does not retry at all. Instead it lets the MapReduce task fail and relies on the Hadoop Job Tracker to retry failed MapReduce task attempts several times according to standard Hadoop semantics. Cloudera Manager can configure the Hadoop Job Tracker to be highly available. On MapReduceIndexerTool startup, all data in the output directory is deleted if that output directory already exists. To retry an entire job that has failed, rerun the program using the same arguments.

For example:

```
hadoop ... MapReduceIndexerTool ... -D isProductionMode=true -D
isIgnoringRecoverableExceptions=true ...
```

Configuring Cloudera Manager for High Availability With a Load Balancer

This section provides an example of configuring Cloudera Manager 5 for high availability using a TCP load balancer. The procedures describe how to configure high availability using a specific, open-source load balancer. Depending on the operational requirements of your CDH deployment, you can select a different load balancer. You can use either a hardware or software load balancer, but must be capable of forwarding all Cloudera Manager ports to backing server instances. (See [Ports Used by Cloudera Manager and Cloudera Navigator](#) for more information about the ports used by Cloudera Manager.)

This topic discusses Cloudera Manager high availability in the context of *active-passive* configurations only; *active-active* configurations are currently unsupported. For more information about the differences between *active-passive* and *active-active* High Availability, see http://en.wikipedia.org/wiki/High-availability_cluster.



Important: Cloudera Support supports all of the configuration and modification to Cloudera software detailed in this document. However, Cloudera Support is unable to assist with issues or failures with the third-party software that is used. Use of any third-party software, or software not directly covered by Cloudera Support, is at the risk of the end user.

Introduction to Cloudera Manager Deployment Architecture

Cloudera Manager consists of the following software components:

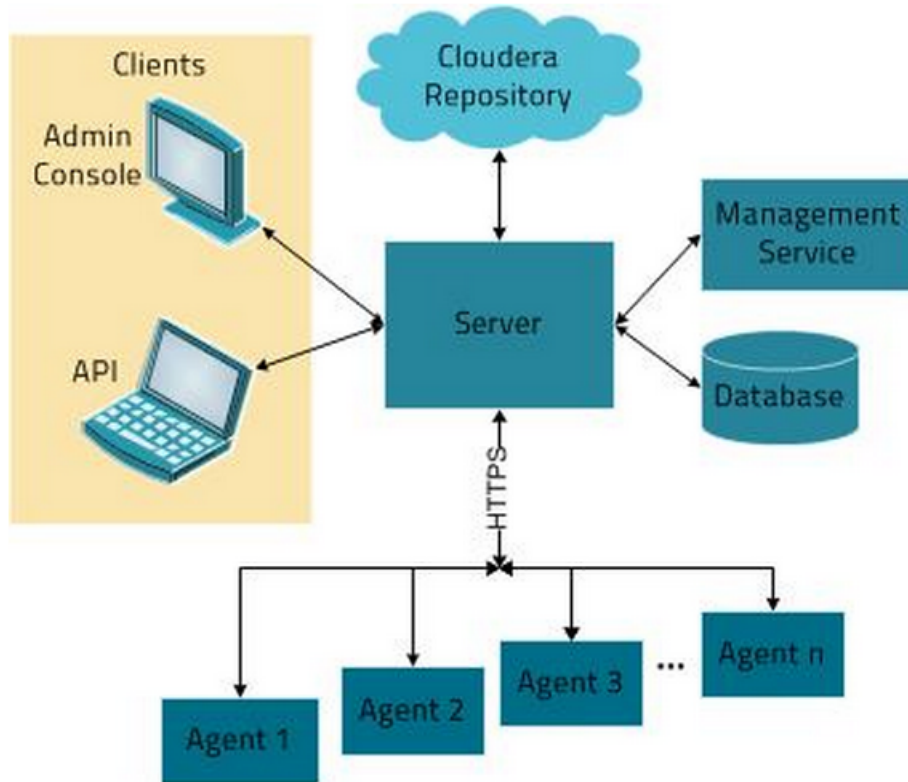


Figure 6: Cloudera Manager Architecture

- Cloudera Manager Server
- Cloudera Management Service
- Relational databases (several)
- Filesystem-based runtime state storage (used by some services that are part of Cloudera Management Service)
- Cloudera Manager Agent (one instance per each managed host)

You can locate the Cloudera Manager Server and Cloudera Management Service on different hosts (with each role of the Cloudera Management Service, such as the Event Server or the Alert Server and so on, possibly located on different hosts).

Cloudera Manager Server and some of the Cloudera Management Service roles (such as Cloudera Navigator) use a relational database to store their operational data. Some other services (such as the Host Monitor and the Service Monitor) use the filesystem (through LevelDB) to store their data.

High availability in the context of Cloudera Manager involves configuring secondary failover instances for each of these services and also for the persistence components (the relational database and the file system) that support these services. For simplicity, this document assumes that all of the Cloudera Management Service roles are located on a single machine.

The Cloudera Manager Agent software includes an *agent* and a *supervisor* process. The agent process handles RPC communication with Cloudera Manager and with the roles of the Cloudera Management Service, and primarily handles configuration changes to your roles. The supervisor process handles the local Cloudera-deployed process lifecycle and handles auto-restarts (if configured) of failed processes.

Prerequisites for Setting up Cloudera Manager High Availability



Note: MySQL GTID-based replication is not supported.

- A multi-homed TCP load balancer, or two TCP load balancers, capable of proxying requests on specific ports to one server from a set of backing servers.
 - The load balancer does not need to support termination of SSL/TLS connections.
 - This load balancer can be hardware or software based, but should be capable of proxying multiple ports. HTTP/HTTPS-based load balancers are insufficient because Cloudera Manager uses several non-HTTP-based protocols internally.
 - This document uses **HAProxy**, a small, open-source, TCP-capable load balancer, to demonstrate a workable configuration.
- A networked storage device that you can configure to be highly available. Typically this is an NFS store, a SAN device, or a storage array that satisfies the read/write throughput requirements of the Cloudera Management Service. This document assumes the use of NFS due to the simplicity of its configuration and because it is an easy, vendor-neutral illustration.
- The procedures in this document require `ssh` access to all the hosts in the cluster where you are enabling high availability for Cloudera Manager.

The Heartbeat Daemon and Virtual IP Addresses

You may have configured Cloudera Manager high availability by configuring virtual IP addresses and using the Heartbeat daemon (<http://linux-ha.org/wiki/Heartbeat>). The original Heartbeat package is deprecated; however, support and maintenance releases are still available through LinBit (<https://www.linbit.com/en/linbit-takes-over-heartbeat-maintenance/>).

Cloudera recommends using Corosync and Pacemaker (both currently maintained through [ClusterLabs](#)). Corosync is an open-source high-availability tool commonly used in the open-source community.

Editions of this document released for Cloudera Manager4 and CDH 4 also used virtual IP addresses that move as a resource from one host to another on failure. Using virtual IP addresses has several drawbacks:

- Questionable reliance on outdated Address Resolution Protocol (ARP) behavior to ensure that the IP-to-MAC translation works correctly to resolve to the new MAC address on failure.
- Split-brain scenarios that lead to problems with routing.
- A requirement that the virtual IP address subnet be shared between the primary and the secondary hosts, which can be onerous if you deploy your secondaries off site.

Therefore, Cloudera no longer recommend the use of virtual IP addresses, and instead recommends using a dedicated load balancer.

Single-User Mode, TLS, and Kerberos

High availability, as described in this document, supports the following:

- Single-user mode. You must run all commands as the `root` user (unless specified otherwise). These procedures do not alter or modify the behavior of how CDH services function.
- TLS and Kerberized deployments. For more information, see [TLS and Kerberos Configuration for Cloudera Manager High Availability](#) on page 375.

High-Level Steps to Configure Cloudera Manager High Availability

To configure Cloudera Manager for high availability, follow these high-level steps. Click each step to see detailed procedures.

Important:
 Unless stated otherwise, run all commands mentioned in this topic as the `root` user.
 You do not need to stop the CDH cluster to configure Cloudera Manager high availability.

- [Step 1: Setting Up Hosts and the Load Balancer](#) on page 350.
- [Step 2: Installing and Configuring Cloudera Manager Server for High Availability](#) on page 355.
- [Step 3: Installing and Configuring Cloudera Management Service for High Availability](#) on page 359.
- [Step 4: Automating Failover with Corosync and Pacemaker](#) on page 365.

Step 1: Setting Up Hosts and the Load Balancer

At a high level, you set up Cloudera Manager Server and Cloudera Management Service *roles* (including Cloudera Navigator) on separate hosts, and make sure that network access to those hosts from other Cloudera services and to the Admin Console occurs through the configured load balancer.

Cloudera Manager Server, Cloudera Navigator, and all of the Cloudera Management Service roles that use a relational database should use an external database server, located off-host. You must make sure that these databases are configured to be highly available. See [Database High Availability Configuration](#) on page 374.

You configure other Cloudera Management Service roles (such as the Service Monitor and Host Monitor roles) that use a file-backed storage mechanism to store their data on a shared NFS storage mechanism.

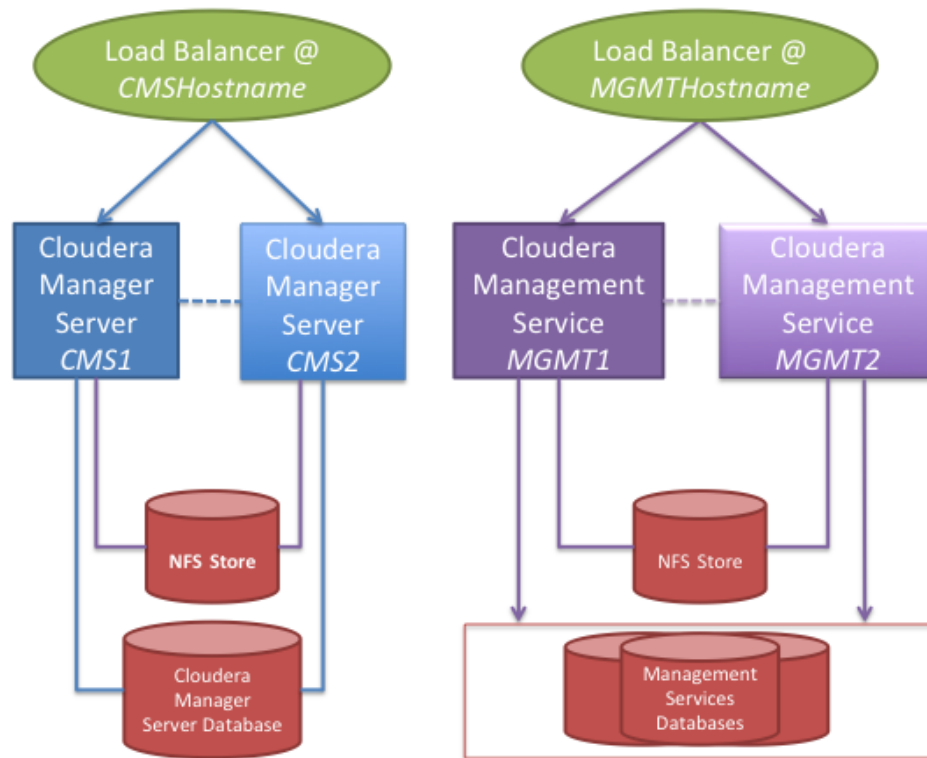


Figure 7: High-level layout of components for Cloudera Manager high availability

Creating Hosts for Primary and Secondary Servers

For this example, Cloudera recommends using four hosts for Cloudera Manager services. All of these hosts must resolve forward and reverse DNS lookups correctly:

- Cloudera Manager Server primary host (hostname: *CMS1*)
- Cloudera Management Service primary host (hostname: *MGMT1*)
- Cloudera Manager Server secondary host (hostname: *CMS2*)
- Cloudera Management Service secondary host (hostname: *MGMT2*)



Note: The hostnames used here are placeholders and are used throughout this document. When configuring your cluster, substitute the actual names of the hosts you use in your environment.

In addition, Cloudera recommends the following:

- Do not host the Cloudera Manager or Cloudera Management Service roles on existing hosts in a CDH cluster, because this complicates failover configuration, and overlapping failure domains can cause problems with fault containment and error tracing.
- Configure both the primary and the secondary hosts using the same host configuration. This helps to ensure that failover does not lead to decreased performance.
- Host the primary and secondary hosts on separate power and network segments within your organization to limit overlapping failure domains.

Setting up the Load Balancer

This procedure demonstrates configuring the load balancer as two separate software load balancers using HAProxy, on two separate hosts for demonstration clarity. (To reduce cost, you might prefer to set up a single load balancer with two network interfaces.) You use one HAProxy host for Cloudera Manager Server and another for the Cloudera Management Service.



Note: HAProxy is used here for demonstration purposes. Production-level performance requirements determine the load balancer that you select for your installation.

HAProxy version 1.5.2 is used for these procedures.

1. Reserve two hostnames in your DNS system, and assign them to each of the load balancer hosts. (The names *CMSHostname*, and *MGMTHostname* are used in this example; substitute the correct hostname for your environment.) These hostnames will be the externally accessible hostnames for Cloudera Manager Server and Cloudera Management Service. (Alternatively, use one load balancer with separate, resolvable IP addresses—one each to back *CMSHostname* and *MGMTHostname* respectively).
 - *CMSHostname* is used to access Cloudera Manager Admin Console.
 - *MGMTHostname* is used for internal access to the Cloudera Management Service from Cloudera Manager Server and Cloudera Manager Agents.
2. Set up two hosts using any supported Linux distribution (RHEL, CentOS, Ubuntu or SUSE; see [Supported Operating Systems](#)) with the hostnames listed above. See the [HAProxy documentation](#) for recommendations on configuring the hardware of these hosts.
3. Install the version of HAProxy that is recommended for the version of Linux installed on the two hosts:

RHEL/CentOS:

```
$ yum install haproxy
```

Ubuntu (use a current Personal Package Archive (PPA) for 1.5 from <http://haproxy.debian.net>):

```
$ apt-get install haproxy
```

SUSE:

```
$ zypper install haproxy
```

4. Configure HAProxy to autostart on both the *CMSHostname* and *MGMTHostname* hosts:

RHEL, CentOS, and SUSE:

```
$ chkconfig haproxy on
```

Ubuntu:

```
$ update-rc.d haproxy defaults
```

5. Configure HAProxy.

- On *CMSHostname*, edit the `/etc/haproxy/haproxy.cfg` files and make sure that the ports listed at [Ports Used by Cloudera Manager and Cloudera Navigator](#) for “Cloudera Manager Server” are proxied. For Cloudera Manager 5, this list includes the following ports as defaults:
 - 7180
 - 7182
 - 7183

Sample HAProxy Configuration for *CMSHostname*

```
listen cmf :7180
  mode tcp
  option tcplog
  server cmfhttp1 CMS1:7180 check
  server cmfhttp2 CMS2:7180 check

listen cmfavro :7182
  mode tcp
  option tcplog
  server cmfavro1 CMS1:7182 check
  server cmfavro2 CMS2:7182 check

#ssl pass-through, without termination
listen cmfhttps :7183
  mode tcp
  option tcplog
  server cmfhttps1 CMS1:7183 check
  server cmfhttps2 CMS2:7183 check
```

- On *MGMTHostname*, edit the `/etc/haproxy/haproxy.cfg` file and make sure that the ports for Cloudera Management Service are proxied (see [Ports Used by Cloudera Manager and Cloudera Navigator](#)). For Cloudera Manager 5, this list includes the following ports as defaults:
 - 5678
 - 7184
 - 7185
 - 7186
 - 7187
 - 8083
 - 8084
 - 8086
 - 8087

- 8091
- 9000
- 9994
- 9995
- 9996
- 9997
- 9998
- 9999
- 10101

Example HAProxy Configuration for *MGMT*Hostname

```

listen mgmt1 :5678
  mode tcp
  option tcplog
  server mgmt1a MGMT1 check
  server mgmt1b MGMT2 check

listen mgmt2 :7184
  mode tcp
  option tcplog
  server mgmt2a MGMT1 check
  server mgmt2b MGMT2 check

listen mgmt3 :7185
  mode tcp
  option tcplog
  server mgmt3a MGMT1 check
  server mgmt3b MGMT2 check
listen mgmt4 :7186
  mode tcp
  option tcplog
  server mgmt4a MGMT1 check
  server mgmt4b MGMT2 check
listen mgmt5 :7187
  mode tcp
  option tcplog
  server mgmt5a MGMT1 check
  server mgmt5b MGMT2 check

listen mgmt6 :8083
  mode tcp
  option tcplog
  server mgmt6a MGMT1 check
  server mgmt6b MGMT2 check
listen mgmt7 :8084
  mode tcp
  option tcplog
  server mgmt7a MGMT1 check
  server mgmt7b MGMT2 check
listen mgmt8 :8086
  mode tcp
  option tcplog
  server mgmt8a MGMT1 check
  server mgmt8b MGMT2 check
listen mgmt9 :8087
  mode tcp
  option tcplog
  server mgmt9a MGMT1 check
  server mgmt9b MGMT2 check
listen mgmt10 :8091
  mode tcp
  option tcplog
  server mgmt10a MGMT1 check
  server mgmt10b MGMT2 check
listen mgmt-agent :9000
  mode tcp
  option tcplog
  server mgmt-agenta MGMT1 check
  server mgmt-agentb MGMT2 check

```

```
listen mgmt11 :9994
  mode tcp
  option tcplog
  server mgmt11a MGMT1 check
  server mgmt11b MGMT2 check
listen mgmt12 :9995
  mode tcp
  option tcplog
  server mgmt12a MGMT1 check
  server mgmt12b MGMT2 check
listen mgmt13 :9996
  mode tcp
  option tcplog
  server mgmt13a MGMT1 check
  server mgmt13b MGMT2 check
listen mgmt14 :9997
  mode tcp
  option tcplog
  server mgmt14a MGMT1 check
  server mgmt14b MGMT2 check
listen mgmt15 :9998
  mode tcp
  option tcplog
  server mgmt15a MGMT1 check
  server mgmt15b MGMT2 check
listen mgmt16 :9999
  mode tcp
  option tcplog
  server mgmt16a MGMT1 check
  server mgmt16b MGMT2 check
listen mgmt17 :10101
  mode tcp
  option tcplog
  server mgmt17a MGMT1 check
  server mgmt17b MGMT2 check
```

After updating the configuration, restart HAProxy on both the *MGMTHostname* and *CMSHostname* hosts:

```
$ service haproxy restart
```

Setting up the Database

1. Create databases on your preferred external database server. See [Cloudera Manager and Managed Service Data Stores](#).



Important: The embedded Postgres database cannot be configured for high availability and should not be used in a high-availability configuration.

2. Configure your databases to be highly available. Consult the vendor documentation for specific information.

MySQL, PostgreSQL, and Oracle each have many options for configuring high availability. See [Database High Availability Configuration](#) on page 374 for some external references on configuring high availability for your Cloudera Manager databases.

Setting up an NFS Server

The procedures outlined for setting up the Cloudera Manager Server and Cloudera Management Service hosts presume there is a shared store configured that can be accessed from both the primary and secondary instances of these hosts. This usually requires that this store be accessible over the network, and can be one of a variety of remote storage mechanisms (such as an iSCSI drive, a SAN array, or an NFS server).



Note: Using NFS as a shared storage mechanism is used here for demonstration purposes. Refer to your Linux distribution documentation on production NFS configuration and security. Production-level performance requirements determine the storage that you select for your installation.

This section describes how to configure an NFS server and assumes that you understand how to configure highly available remote storage devices. Further details are beyond the scope and intent of this guide.

There are no intrinsic limitations on where this NFS server is located, but because overlapping failure domains can cause problems with fault containment and error tracing, Cloudera recommends that you not co-locate the NFS server with any CDH or Cloudera Manager servers or the load-balancer hosts detailed in this document.

1. Install NFS on your designated server:

RHEL/CentOS

```
$ yum install nfs-utils nfs-utils-lib
```

Ubuntu

```
$ apt-get install nfs-kernel-server
```

SUSE

```
$ zypper install nfs-kernel-server
```

2. Start `nfs` and `rpcbind`, and configure them to autostart:

RHEL/CentOS:

```
$ chkconfig nfs on
$ service rpcbind start
$ service nfs start
```

Ubuntu:

```
$ update-rc.d nfs defaults
$ service rpcbind start
$ service nfs-kernel-server
```

SUSE:

```
$ chkconfig nfs on
$ service rpcbind start
$ service nfs-kernel-server start
```



Note: Later sections describe mounting the shared directories and sharing them between the primary and secondary instances.

Step 2: Installing and Configuring Cloudera Manager Server for High Availability

You can use an existing Cloudera Manager installation and extend it to a high-availability configuration, as long as you are not using the embedded PostgreSQL database.

This section describes how to install and configure a failover secondary for Cloudera Manager Server that can take over if the primary fails.

This section does not cover installing instances of Cloudera Manager Agent on `CMS1` or `CMS2` and configuring them to be highly available. See [Installing Cloudera Manager and CDH](#).

Setting up NFS Mounts for Cloudera Manager Server

1. Create the following directories on the NFS server [you created in a previous step](#):

```
$ mkdir -p /media/cloudera-scm-server
```

2. Mark these mounts by adding these lines to the `/etc/exports` file on the NFS server:

```
/media/cloudera-scm-server CMS1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-server CMS2(rw,sync,no_root_squash,no_subtree_check)
```

3. Export the mounts by running the following command on the NFS server:

```
$ exportfs -a
```

4. Set up the filesystem mounts on `CMS1` and `CMS2` hosts:

- a. If you are updating an existing installation for high availability, stop the Cloudera Manager Server if it is running on either of the `CMS1` or `CMS2` hosts by running the following command:

```
$ service cloudera-scm-server stop
```

- b. Make sure that the NFS mount helper is installed:

RHEL/CentOS:

```
$ yum install nfs-utils-lib
```

Ubuntu:

```
$ apt-get install nfs-common
```

SUSE:

```
$ zypper install nfs-client
```

- c. Make sure that `rpcbind` is running and has been restarted:

```
$ service rpcbind restart
```

5. Create the mount points on both `CMS1` and `CMS2`:

- a. If you are updating an existing installation for high availability, copy the `/var/lib/cloudera-scm-server` file from your existing Cloudera Manager Server host to the NFS server with the following command (`NFS` refers to the NFS server you created in a previous step):

```
$ scp -r /var/lib/cloudera-scm-server/ NFS:/media/cloudera-scm-server
```

- b. Set up the `/var/lib/cloudera-scm-server` directory on the `CMS1` and `CMS2` hosts:

```
$ rm -rf /var/lib/cloudera-scm-server
$ mkdir -p /var/lib/cloudera-scm-server
```

- c. Mount the following directory to the NFS mounts, on both `CMS1` and `CMS2`:

```
$ mount -t nfs NFS:/media/cloudera-scm-server /var/lib/cloudera-scm-server
```

- d. Set up `fstab` to persist the mounts across restarts by editing the `/etc/fstab` file on `CMS1` and `CMS2` and adding the following lines:

```
NFS:/media/cloudera-scm-server /var/lib/cloudera-scm-server nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
```

Installing the Primary

Updating an Existing Installation for High Availability

You can retain your existing Cloudera Manager Server as-is, if the deployment meets the following conditions:

- The Cloudera Management Service is located on a single host that is not the host where Cloudera Manager Server runs.
- The data directories for the roles of the Cloudera Management Service are located on a remote storage device (such as an NFS store), and they can be accessed from both primary and secondary installations of the Cloudera Management Service.

If your deployment does not meet these conditions, Cloudera recommends that you uninstall Cloudera Management Services by stopping the existing service and deleting it.



Important: Deleting the Cloudera Management Service leads to loss of all existing data from the Host Monitor and Service Monitor roles that store health and monitoring information for your cluster on the local disk associated with the host(s) where those roles are installed.

To delete and remove the Cloudera Management Service:

1. Open the Cloudera Manager Admin Console and go to the **Home** page.
2. Click **Cloudera Management Service > Stop**.
3. Click **Cloudera Management Service > Delete**.

Fresh Installation

Use either of the installation paths (B or C) specified in the documentation to install Cloudera Manager Server, but do not add “Cloudera Management Service” to your deployment until you complete [Step 3: Installing and Configuring Cloudera Management Service for High Availability](#) on page 359, which describes how to set up the Cloudera Management Service.

See:

- [Installation Path B - Manual Installation Using Cloudera Manager Packages](#)
- [Installation Path C - Manual Installation Using Cloudera Manager Tarballs](#)

You can now start the freshly-installed Cloudera Manager Server on `CMS1`:

```
$ service cloudera-scm-server start
```

Before proceeding, verify that you can access the Cloudera Manager Admin Console at `http://CMS1:7180`.

If you have just installed Cloudera Manager, click the Cloudera Manager logo to skip adding new hosts and to gain access to the Administration menu, which you need for the following steps.

HTTP Referer Configuration

Cloudera recommends that you disable the HTTP Referer check because it causes problems for some proxies and load balancers. Check the configuration manual of your proxy or load balancer to determine if this is necessary.

To disable HTTP Referer in the Cloudera Manager Admin Console:

1. Select **Administration > Settings**.
2. Select **Category > Security**.
3. Deselect the **HTTP Referer Check** property.

Before proceeding, verify that you can access the Cloudera Manager Admin Console through the load balancer at `http://CMSHostname:7180`.

TLS and Kerberos Configuration

To configure Cloudera Manager to use TLS encryption or authentication, or to use Kerberos authentication, see [TLS and Kerberos Configuration for Cloudera Manager High Availability](#) on page 375.

Installing the Secondary

Setting up the Cloudera Manager Server secondary requires copying certain files from the primary to ensure that they are consistently initialized.

1. On the `CMS2` host, install the `cloudera-manager-server` package using Installation Path B or Installation Path C.

See:

- [Installation Path B - Manual Installation Using Cloudera Manager Packages](#)
- [Installation Path C - Manual Installation Using Cloudera Manager Tarballs](#)

2. When setting up the database on the secondary, copy the `/etc/cloudera-scm-server/db.properties` file from host `CMS1` to host `CMS2` at `/etc/cloudera-scm-server/db.properties`. For example:

```
$ mkdir -p /etc/cloudera-scm-server
$ scp [<ssh-user>@]CMS1:/etc/cloudera-scm-server/db.properties
/etc/cloudera-scm-server/db.properties
```

3. If you configured Cloudera Manager TLS encryption or authentication, or Kerberos authentication in your primary installation, see [TLS and Kerberos Configuration for Cloudera Manager High Availability](#) on page 375 for additional configuration steps.
4. Do not start the `cloudera-scm-server` service on this host yet, and disable autostart on the secondary to avoid automatically starting the service on this host.

RHEL/CentOS/SUSEL:

```
$ chkconfig cloudera-scm-server off
```

Ubuntu:

```
$ update-rc.d -f cloudera-scm-server remove
```

(You will also disable autostart on the primary when you configure [automatic failover](#) in a later step.) Data corruption can result if both primary and secondary Cloudera Manager Server instances are running at the same time, and it is not supported. :

Testing Failover

Test failover manually by using the following steps:

1. Stop `cloudera-scm-server` on your primary host (`CMS1`):

```
$ service cloudera-scm-server stop
```

2. Start `cloudera-scm-server` on your secondary host (`CMS2`):

```
$ service cloudera-scm-server start
```

3. Wait a few minutes for the service to load, and then access the Cloudera Manager Admin Console through a web browser, using the load-balanced hostname (for example: `http://CMSHostname:CMS_port`).

Now, fail back to the primary before configuring the Cloudera Management Service on your installation:

1. Stop `cloudera-scm-server` on your secondary machine (*CMS2*):

```
$ service cloudera-scm-server stop
```

2. Start `cloudera-scm-server` on your primary machine (*CMS1*):

```
$ service cloudera-scm-server start
```

3. Wait a few minutes for the service to load, and then access the Cloudera Manager Admin Console through a web browser, using the load-balanced hostname (for example: `http://CMSHostname:7180`).

Updating Cloudera Manager Agents to use the Load Balancer

After completing the primary and secondary installation steps listed previously, update the Cloudera Manager Agent configuration on all of the hosts associated with this Cloudera Manager installation, except the *MGMT1*, *MGMT2*, *CMS1*, and *CMS2* hosts, to use the load balancer address:

1. Connect to a shell on each host where CDH processes are installed and running. (The *MGMT1*, *MGMT2*, *CMS1*, and *CMS2* hosts do not need to be modified as part of this step.)
2. Update the `/etc/cloudera-scm-agent/config.ini` file and change the `server_host` line:

```
server_host = <CMSHostname>
```

3. Restart the agent (this command starts the agents if they are not running):

```
$ service cloudera-scm-agent restart
```

Step 3: Installing and Configuring Cloudera Management Service for High Availability

This section demonstrates how to set up shared mounts on *MGMT1* and *MGMT2*, and then install Cloudera Management Service to use those mounts on the primary and secondary servers.



Important: Do not start the primary and secondary servers that are running Cloudera Management Service at the same time. Data corruption can result.

Setting up NFS Mounts for Cloudera Management Service

1. Create directories on the NFS server:

```
$ mkdir -p /media/cloudera-host-monitor
$ mkdir -p /media/cloudera-scm-agent
$ mkdir -p /media/cloudera-scm-eventserver
$ mkdir -p /media/cloudera-scm-headlamp
$ mkdir -p /media/cloudera-service-monitor
$ mkdir -p /media/cloudera-scm-navigator
$ mkdir -p /media/etc-cloudera-scm-agent
```

2. Mark these mounts by adding the following lines to the `/etc/exports` file on the NFS server:

```
/media/cloudera-host-monitor MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-agent MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-eventserver MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-headlamp MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-navigator MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/etc-cloudera-scm-agent MGMT1(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-host-monitor MGMT2(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-agent MGMT2(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-eventserver MGMT2(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-scm-headlamp MGMT2(rw,sync,no_root_squash,no_subtree_check)
/media/cloudera-service-monitor MGMT2(rw,sync,no_root_squash,no_subtree_check)
```

```
/media/cloudera-scm-navigator MGMT2(rw, sync, no_root_squash, no_subtree_check)
/media/etc-cloudera-scm-agent MGMT2(rw, sync, no_root_squash, no_subtree_check)
```

3. Export the mounts running the following command on the NFS server:

```
$ exportfs -a
```

4. Set up the filesystem mounts on *MGMT1* and *MGMT2* hosts:

a. Make sure that the NFS mount helper is installed:

RHEL/CentOS:

```
$ yum install nfs-utils-lib
```

Ubuntu:

```
$ apt-get install nfs-common
```

SUSE:

```
$ zypper install nfs-client
```

b. Create the mount points on both *MGMT1* and *MGMT2*:

```
$ mkdir -p /var/lib/cloudera-host-monitor
$ mkdir -p /var/lib/cloudera-scm-agent
$ mkdir -p /var/lib/cloudera-scm-eventserver
$ mkdir -p /var/lib/cloudera-scm-headlamp
$ mkdir -p /var/lib/cloudera-service-monitor
$ mkdir -p /var/lib/cloudera-scm-navigator
$ mkdir -p /etc/cloudera-scm-agent
```

c. Mount the following directories to the NFS mounts, on both *MGMT1* and *MGMT2* (*NFS* refers to the server NFS hostname or IP address):

```
$ mount -t nfs NFS:/media/cloudera-host-monitor /var/lib/cloudera-host-monitor
$ mount -t nfs NFS:/media/cloudera-scm-agent /var/lib/cloudera-scm-agent
$ mount -t nfs NFS:/media/cloudera-scm-eventserver /var/lib/cloudera-scm-eventserver
$ mount -t nfs NFS:/media/cloudera-scm-headlamp /var/lib/cloudera-scm-headlamp
$ mount -t nfs NFS:/media/cloudera-service-monitor /var/lib/cloudera-service-monitor
$ mount -t nfs NFS:/media/cloudera-scm-navigator /var/lib/cloudera-scm-navigator
$ mount -t nfs NFS:/media/etc-cloudera-scm-agent /etc/cloudera-scm-agent
```

5. Set up *fstab* to persist the mounts across restarts. Edit the */etc/fstab* file and add these lines:

```
NFS:/media/cloudera-host-monitor /var/lib/cloudera-host-monitor nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/cloudera-scm-agent /var/lib/cloudera-scm-agent nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/cloudera-scm-eventserver /var/lib/cloudera-scm-eventserver nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/cloudera-scm-headlamp /var/lib/cloudera-scm-headlamp nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/cloudera-scm-service-monitor /var/lib/cloudera-service-monitor nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/cloudera-scm-navigator /var/lib/cloudera-scm-navigator nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
NFS:/media/etc-cloudera-scm-agent /etc/cloudera-scm-agent nfs
auto,noatime,nolock,intr,tcp,actimeo=1800 0 0
```


Installing the Primary

1. Connect to a shell on *MGMT1*, and then install the `cloudera-manager-daemons` and `cloudera-manager-agent` packages:
 - a. Install packages `cloudera-manager-daemons` and `cloudera-manager-agent` packages using instructions from Installation Path B (See [Installation Path B - Manual Installation Using Cloudera Manager Packages](#)).
 - b. Install the Oracle Java JDK version that is required for your deployment, if it is not already installed on the host. See [Supported JDK Versions](#).
2. Configure the agent to report its hostname as `<MGMTHostname>` to Cloudera Manager. This ensures that the connections from the Cloudera Manager Agents on the CDH cluster hosts report to the correct Cloudera Management Service host in the event of a failover.

- a. Edit the `/etc/cloudera-scm-agent/config.ini` file to update the following lines:

```
server_host=CMSHostname
listening_hostname=MGMTHostname
```

- b. Edit the `/etc/hosts` file and add `MGMTHostname` as an alias for your public IP address for *MGMT1* by adding a line like this at the end of your `/etc/hosts` file:

```
MGMT1 IP MGMTHostname
```

- c. Confirm that the alias has taken effect by running the `ping` command. For example:

```
[root@MGMT1 ~]# ping MGMTHostname
PING MGMTHostname (MGMT1 IP) 56(84) bytes of data.
64 bytes from MGMTHostname (MGMT1 IP): icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from MGMTHostname (MGMT1 IP): icmp_seq=2 ttl=64 time=0.018 ms
...
```

- d. Make sure that the `cloudera-scm` user and the `cloudera-scm` group have access to the mounted directories under `/var/lib`, by using the `chown` command on `cloudera-scm`. For example, run the following on *MGMT1*:

```
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-scm-eventserver
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-scm-navigator
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-service-monitor
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-host-monitor
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-scm-agent
$ chown -R cloudera-scm:cloudera-scm /var/lib/cloudera-scm-headlamp
```



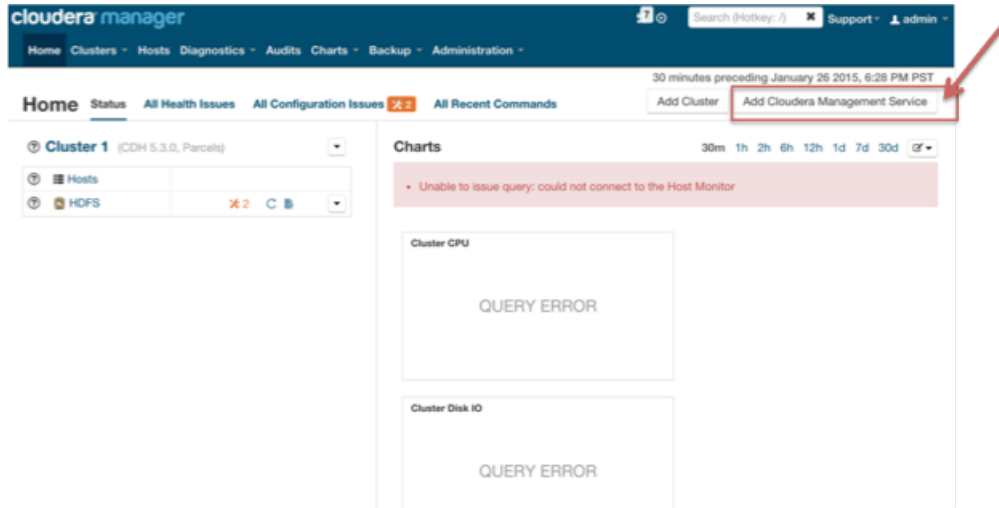
Note: The `cloudera-scm` user and the `cloudera-scm` group are the default owners as specified in Cloudera Management Service advanced configuration. If you alter these settings, or are using [single-user mode](#), modify the above `chown` instructions to use the altered user or group name.

- e. Restart the agent on *MGMT1* (this also starts the agent if it is not running):

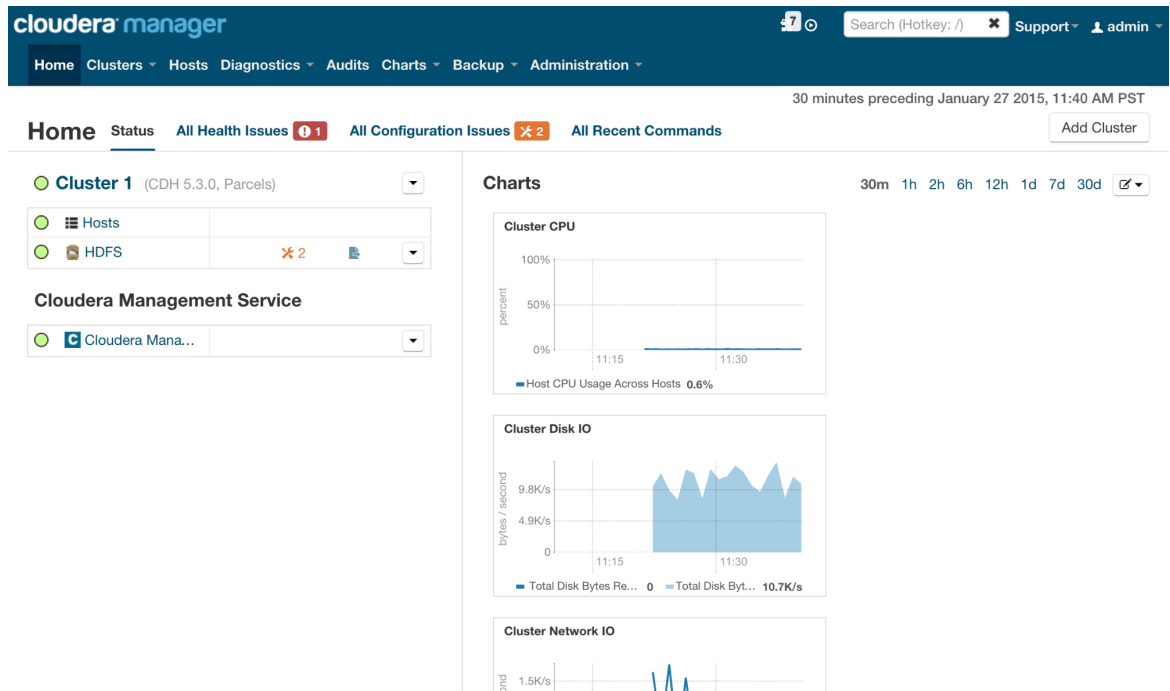
```
$ service cloudera-scm-agent restart
```

- f. Connect to the Cloudera Manager Admin Console running on `<CMSHostname>` and:

- a. Go to the **Hosts** tab and make sure that a host with name `<MGMTHostname>` is reported. (If it is not available yet, wait for it to show up before you proceed.)
- b. Click **Add Cloudera Management Service**.



- g. Make sure you install all of the roles of the Cloudera Management Service on the host named *MGMTHost.name*.
- h. Proceed through the steps to configure the roles of the service to use your database server, and use defaults for the storage directory for Host Monitor or Service Monitor.
- i. After you have completed the steps, wait for the Cloudera Management Service to finish starting, and verify the health status of your clusters as well as the health of the Cloudera Management Service as reported in the Cloudera Manager Admin Console. The health status indicators should be green, as shown:



The service health for Cloudera Management Service might, however, show as red:

Quick Links

Event Search [Alerts](#) [Critical](#) [All](#)

Status Summary

Event Server	● Bad Health
Host Monitor	● Bad Health
Navigator Metadata Server	● Bad Health
Activity Monitor	● Bad Health
Reports Manager	● Bad Health
Navigator Audit Server	● Bad Health
Service Monitor	● Bad Health
Alert Publisher	● Bad Health

Health Tests [Collapse All](#)

▼ ● 8 bad.

- The health of the Activity Monitor is bad. The following health tests are bad: host health. [Details](#)
- The health of the Service Monitor is bad. The following health tests are bad: host health. [Details](#)
- The health of the Host Monitor is bad. The following health tests are bad: host health. [Details](#)

Charts

30m 1h 2h 6h 12h 1d 7d 30d

CPU Cores Used

cores

12 PM 12:15

ACTIVITYMONI... 0.01 ALERTPUBLIS... 0.01
EVENTSERVER... 0.03 HOSTMONITO... 0.03

Health

percent

12 PM 12:15

bad health 100 concerning health 0
disabled health 0 good health 0

Important Events and Alerts

events

In this case, you need to identify whether the health test failure is caused by the **Hostname and Canonical Name Health Check** for the `MGMTHostname` host, which might look like this:

Health Tests [Expand All](#)

● The hostname and canonical name for this host are [Details](#) not consistent when checked from a Java process.

This test can fail in this way because of the way you modified `/etc/hosts` on `MGMT1` and `MGMT2` to allow the resolution of `MGMTHostname` locally. This test can be safely disabled on the `MGMTHostname` host from the Cloudera Manager Admin Console.

- j. If you are configuring Kerberos and SSL, see [TLS and Kerberos Configuration for Cloudera Manager High Availability](#) on page 375 for configuration changes as part of this step.

Installing the Secondary

1. Stop all Cloudera Management Service roles using the Cloudera Manager Admin Console:

- a. On the Home page, click



to the right of **Cloudera Management Service** and select **Stop**.

- b. Click **Stop** to confirm. The **Command Details** window shows the progress of stopping the roles.

- c. When **Command completed with n/n successful subcommands** appears, the task is complete. Click **Close**.

2. Stop the `cloudera-scm-agent` service on `MGMT1`:

```
$ service cloudera-scm-agent stop
```

3. Connect to a shell on `MGMT2`, and then install `cloudera-manager-daemons` and `cloudera-manager-agent`:

- a. Install the `cloudera-manager-daemons` and `cloudera-manager-agent` packages using instructions from Installation Path B. See [Installation Path B - Manual Installation Using Cloudera Manager Packages](#).

- b. Install the Oracle Java JDK version that is required for your deployment, if it is not already installed on the host. See [Supported JDK Versions](#).
- 4. Configure the agent to report its hostname as `MGMTHostname` to Cloudera Manager, as described previously in [Installing the Primary](#) on page 361.
 - a. Make sure that `/etc/cloudera-scm-agent/config.ini` has the following lines (because this is a shared mount with the primary, it should be the same as in the primary installation):

```
server_host=<CMHostname>
listening_hostname=<MGMTHostname>
```

- b. Edit the `/etc/hosts` file and add `MGMTHostname` as an alias for your public IP address for `MGMT2`, by adding a line like this at the end of your `/etc/hosts` file:

```
<MGMT2-IP> <MGMTHostname>
```

- c. Confirm that the alias is working by running the `ping` command. For example:

```
[root@MGMT2 ~]# ping MGMTHostname
PING MGMTHostname (MGMT2 IP) 56(84) bytes of data.
64 bytes from MGMTHostname (MGMT2 IP): icmp_seq=1 ttl=64 time=0.034 ms
64 bytes from MGMTHostname (MGMT2 IP): icmp_seq=2 ttl=64 time=0.018 ms
```

- 5. Start the agent on `MGMT2` by running the following command:

```
$ service cloudera-scm-agent start
```

- 6. Log into the Cloudera Manager Admin Console in a web browser and start all Cloudera Management Service roles.

This starts the Cloudera Management Service on `MGMT2`.

- a. Wait for the Cloudera Manager Admin Console to report that the services have started.
- b. Confirm that the services have started on this host by running the following command on `MGMT2`:

```
$ ps -elf | grep "scm"
```

You should see ten total processes running on that host, including the eight Cloudera Management Service processes, a Cloudera Manager Agent process, and a Supervisor process.

- c. Test the secondary installation through the Cloudera Management Admin Console, and inspect the health of the Cloudera Management Service roles, before proceeding.



Note:

Make sure that the UID and GID for the `cloudera-scm` user on the primary and secondary Cloudera Management Service hosts are same; this ensures that the correct permissions are available on the shared directories after failover.

Failing Back to the Primary

Before finishing the installation, fail back to the primary host (`MGMT1`):

- 1. Stop the `cloudera-scm-agent` service on `MGMT2`:

```
$ service cloudera-scm-agent hard_stop_confirmed
```

2. Start the `cloudera-scm-agent` service on `MGMT1`:

```
$ service cloudera-scm-agent start
```

Step 4: Automating Failover with Corosync and Pacemaker

[Corosync](#) and [Pacemaker](#) are popular high-availability utilities that allow you to configure Cloudera Manager to fail over automatically.

This document describes one way to set up clustering using these tools. Actual setup can be done in several ways, depending on the network configuration of your environment.

Prerequisites:

1. Install Pacemaker and Corosync on `CMS1`, `MGMT1`, `CMS2`, and `MGMT2`, using the correct versions for your Linux distribution:



Note: The versions referred to for setting up automatic failover in this document are Pacemaker 1.1.11 and Corosync 1.4.7. See <http://clusterlabs.org/wiki/Install> to determine what works best for your Linux distribution.

RHEL/CentOS:

```
$ yum install pacemaker corosync
```

Ubuntu:

```
$ apt-get install pacemaker corosync
```

SUSE:

```
$ zypper install pacemaker corosync
```

2. Make sure that the `crm` tool exists on all of the hosts. This procedure uses the `crm` tool, which works with Pacemaker configuration. If this tool is not installed when you installed Pacemaker (verify this by running `which crm`), you can download and install the tool for your distribution using the instructions at <http://crmsh.github.io/installation>.

About Corosync and Pacemaker

- By default, Corosync and Pacemaker are not autostarted as part of the boot sequence. Cloudera recommends leaving this as is. If the machine crashes and restarts, manually make sure that failover was successful and determine the cause of the restart before manually starting these processes to achieve higher availability.
 - If the `/etc/default/corosync` file exists, make sure that `START` is set to `yes` in that file:

```
START=yes
```

- Make sure that Corosync is not set to start automatically, by running the following command:

RHEL/CentOS/SUSE:

```
$ chkconfig corosync off
```

Ubuntu:

```
$ update-rc.d -f corosync remove
```

- Note which version of Corosync is installed. The contents of the configuration file for Corosync (`corosync.conf`) that you edit varies based on the version suitable for your distribution. Sample configurations are supplied in this document and are labeled with the Corosync version.

- This document does not demonstrate configuring Corosync with authentication (with `secauth` set to `on`). The Corosync website demonstrates a mechanism to encrypt traffic using symmetric keys.
- Firewall configuration:

Corosync uses UDP transport on ports 5404 and 5405, and these ports must be open for both inbound and outbound traffic on all hosts. If you are using IP tables, run a command similar to the following:

```
$ sudo iptables -I INPUT -m state --state NEW -p udp -m multiport --dports 5404,5405 -j ACCEPT
$ sudo iptables -I OUTPUT -m state --state NEW -p udp -m multiport --sports 5404,5405 -j ACCEPT
```

Setting up Cloudera Manager Server

Set up a Corosync cluster over unicast, between `CMS1` and `CMS2`, and make sure that the hosts can “cluster” together. Then, set up Pacemaker to register Cloudera Manager Server as a resource that it monitors and to fail over to the secondary when needed.

Setting up Corosync

1. Edit the `/etc/corosync/corosync.conf` file on `CMS1` and replace the entire contents with the following text (use the correct version for your environment):

Corosync version 1.x:

```
compatibility: whitetank
totem {
    version: 2
    secauth: off
    interface {
        member {
            memberaddr: CMS1
        }
        member {
            memberaddr: CMS2
        }
        ringnumber: 0
        bindnetaddr: CMS1
        mcastport: 5405
    }
    transport: udpu
}

logging {
    fileline: off
    to_logfile: yes
    to_syslog: yes
    logfile: /var/log/cluster/corosync.log
    debug: off
    timestamp: on
    logger_subsys {
        subsys: AMF
        debug: off
    }
}

service {
    # Load the Pacemaker Cluster Resource Manager
    name: pacemaker
    ver: 1
    #
}
```

Corosync version 2.x:

```
totem {
    version: 2
    secauth: off
    cluster_name: cmf
    transport: udpu
}
```

```

}

nodelist {
  node {
    ring0_addr: CMS1
    nodeid: 1
  }
  node {
    ring0_addr: CMS2
    nodeid: 2
  }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

```

2. Edit the `/etc/corosync/corosync.conf` file on `CMS2`, and replace the entire contents with the following text (use the correct version for your environment):

Corosync version 1.x:

```

compatibility: whitetank
totem {
  version: 2
  secauth: off
  interface {
    member {
      memberaddr: CMS1
    }
    member {
      memberaddr: CMS2
    }
    ringnumber: 0
    bindnetaddr: CMS2
    mcastport: 5405
  }
  transport: udpu
}

logging {
  fileline: off
  to_logfile: yes
  to_syslog: yes
  logfile: /var/log/cluster/corosync.log
  debug: off
  timestamp: on
  logger_subsys {
    subsys: AMF
    debug: off
  }
}

service {
  # Load the Pacemaker Cluster Resource Manager
  name: pacemaker
  ver: 1
  #
}

```

Corosync version 2.x:

```

totem {
  version: 2
  secauth: off
  cluster_name: cmf
  transport: udpu
}

nodelist {
  node {

```

```

        ring0_addr: CMS1
        nodeid: 1
    }
    node {
        ring0_addr: CMS2
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

```

3. Restart Corosync on *CMS1* and *CMS2* so that the new configuration takes effect:

```
$ service corosync restart
```

Setting up Pacemaker

You use Pacemaker to set up Cloudera Manager Server as a *cluster resource*.

See the Pacemaker configuration reference at

http://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/ for more details about Pacemaker options.

The following steps demonstrate one way, recommended by Cloudera, to configure Pacemaker for simple use:

1. Disable autostart for Cloudera Manager Server (because you manage its lifecycle through Pacemaker) on both *CMS1* and *CMS2*:

RHEL/CentOS/SUSE:

```
$ chkconfig cloudera-scm-server off
```

Ubuntu:

```
$ update-rc.d -f cloudera-scm-server remove
```

2. Make sure that Pacemaker has been started on both *CMS1* and *CMS2*:

```
$ /etc/init.d/pacemaker start
```

3. Make sure that *crm* reports two nodes in the cluster:

```

# crm status
Last updated: Wed Mar  4 18:55:27 2015
Last change: Wed Mar  4 18:38:40 2015 via crmd on CMS1
Stack: corosync
Current DC: CMS1 (1) - partition with quorum
Version: 1.1.10-42f2063
2 Nodes configured
0 Resources configured

```

4. Change the Pacemaker cluster configuration (on either *CMS1* or *CMS2*):

```

$ crm configure property no-quorum-policy=ignore
$ crm configure property stonith-enabled=false
$ crm configure rsc_defaults resource-stickiness=100

```

These commands do the following:

- Disable quorum checks. (Because there are only two nodes in this cluster, quorum cannot be established.)
- Disable STONITH explicitly (see [Enabling STONITH \(Shoot the other node in the head\)](#) on page 369).

- Reduce the likelihood of the resource being moved among hosts on restarts.

5. Add Cloudera Manager Server as an LSB-managed resource (either on *CMS1* or *CMS2*):

```
$ crm configure primitive cloudera-scm-server lsb:cloudera-scm-server
```

6. Verify that the primitive has been picked up by Pacemaker:

```
$ crm_mon
```

For example:

```
$ crm_mon
Last updated: Tue Jan 27 15:01:35 2015
Last change: Mon Jan 27 14:10:11 2015
Stack: classic openais (with plugin)
Current DC: CMS1 - partition with quorum
Version: 1.1.11-97629de
2 Nodes configured, 2 expected votes
1 Resources configured
Online: [ CMS1 CMS2 ]
cloudera-scm-server (lsb:cloudera-scm-server): Started CMS1
```

At this point, Pacemaker manages the status of the `cloudera-scm-server` service on hosts *CMS1* and *CMS2*, ensuring that only one instance is running at a time.



Note: Pacemaker expects all lifecycle actions, such as start and stop, to go through Pacemaker; therefore, running `direct service start` or `service stop` commands breaks that assumption.

Testing Failover with Pacemaker

Test Pacemaker failover by running the following command to move the `cloudera-scm-server` resource to *CMS2*:

```
$ crm resource move cloudera-scm-server <CMS2>
```

Test the resource move by connecting to a shell on *CMS2* and verifying that the `cloudera-scm-server` process is now active on that host. It takes usually a few minutes for the new services to come up on the new host.

Enabling STONITH (Shoot the other node in the head)

The following link provides an explanation of the problem of fencing and ensuring (within reasonable limits) that only one host is running a shared resource at a time:

http://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html-single/Clusters_from_Scratch/index.html#idm140603947390416

As noted in that link, you can use several methods (such as [IPMI](#)) to achieve reasonable guarantees on remote host shutdown. Cloudera recommends enabling STONITH, based on the hardware configuration in your environment.

Setting up the Cloudera Manager Service

Setting Up Corosync

1. Edit the `/etc/corosync/corosync.conf` file on *MGMT1* and replace the entire contents with the contents below; make sure to use the correct section for your version of Corosync:

Corosync version 1.x:

```
compatibility: whitetank
totem {
    version: 2
    secauth: off
    interface {
        member {
            memberaddr: MGMT1
        }
        member {
```

```

                memberaddr: MGMT2
            }
            ringnumber: 0
            bindnetaddr: MGMT1
            mcastport: 5405
        }
        transport: udpu
    }
}

logging {
    fileline: off
    to_logfile: yes
    to_syslog: yes
    logfile: /var/log/cluster/corosync.log
    debug: off
    timestamp: on
    logger_subsys {
        subsys: AMF
        debug: off
    }
}

service {
    # Load the Pacemaker Cluster Resource Manager
    name: pacemaker
    ver: 1
    #
}

```

Corosync version 2.x:

```

totem {
    version: 2
    secauth: off
    cluster_name: mgmt
    transport: udpu
}

nodelist {
    node {
        ring0_addr: MGMT1
        nodeid: 1
    }
    node {
        ring0_addr: MGMT2
        nodeid: 2
    }
}

quorum {
    provider: corosync_votequorum
    two_node: 1
}

```

2. Edit the `/etc/corosync/corosync.conf` file on *MGMT2* and replace the contents with the contents below:

Corosync version 1.x:

```

compatibility: whitetank
totem {
    version: 2
    secauth: off
    interface {
        member {
            memberaddr: MGMT1
        }
        member {
            memberaddr: MGMT2
        }
    }
    ringnumber: 0
    bindnetaddr: MGMT2
    mcastport: 5405
}

```

```

    }
    transport: udpu
}
logging {
  fileline: off
  to_logfile: yes
  to_syslog: yes
  logfile: /var/log/cluster/corosync.log
  debug: off
  timestamp: on
  logger_subsys {
    subsys: AMF
    debug: off
  }
}
service {
  # Load the Pacemaker Cluster Resource Manager
  name: pacemaker
  ver: 1
  #
}

```

Corosync version 2.x:

```

totem {
  version: 2
  secauth: off
  cluster_name: mgmt
  transport: udpu
}

nodelist {
  node {
    ring0_addr: CMS1
    nodeid: 1
  }
  node {
    ring0_addr: CMS2
    nodeid: 2
  }
}

quorum {
  provider: corosync_votequorum
  two_node: 1
}

```

3. Restart Corosync on *MGMT1* and *MGMT2* for the new configuration to take effect:

```
$ service corosync restart
```

4. Test whether Corosync has set up a cluster, by using the `corosync-cmapctl` or `corosync-objctl` commands. You should see two members with status `joined`:

```

corosync-objctl | grep "member"
runtime.totem.pg.mrp.srp.members.1.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.1.ip (str) = r(0) ip(MGMT1)
runtime.totem.pg.mrp.srp.members.1.join_count (u32) = 1
runtime.totem.pg.mrp.srp.members.1.status (str) = joined
runtime.totem.pg.mrp.srp.members.2.config_version (u64) = 0
runtime.totem.pg.mrp.srp.members.2.ip (str) = r(0) ip(MGMT2)
runtime.totem.pg.mrp.srp.members.2.join_count (u32) = 1
runtime.totem.pg.mrp.srp.members.2.status (str) = joined

```

Setting Up Pacemaker

Use Pacemaker to set up Cloudera Management Service as a *cluster resource*.

See the Pacemaker configuration reference at

http://clusterlabs.org/pacemaker/doc/en-US/Pacemaker/1.1/html/Clusters_from_Scratch/ for more information about Pacemaker options.

Because the lifecycle of Cloudera Management Service is managed through the Cloudera Manager Agent, you configure the Cloudera Manager Agent to be highly available.

Follow these steps to configure Pacemaker, recommended by Cloudera for simple use:

1. Disable autostart for the Cloudera Manager Agent (because Pacemaker manages its lifecycle) on both *MGMT1* and *MGMT2*:

RHEL/CentOS/SUSE

```
$ chkconfig cloudera-scm-agent off
```

Ubuntu:

```
$ update-rc.d -f cloudera-scm-agent remove
```

2. Make sure that Pacemaker is started on both *MGMT1* and *MGMT2*:

```
$ /etc/init.d/pacemaker start
```

3. Make sure that the `crm` command reports two nodes in the cluster; you can run this command on either host:

```
# crm status
Last updated: Wed Mar  4 18:55:27 2015
Last change: Wed Mar  4 18:38:40 2015 via crmd on MGMT1
Stack: corosync
Current DC: MGMT1 (1) - partition with quorum
Version: 1.1.10-42f2063
2 Nodes configured
0 Resources configured
```

4. Change the Pacemaker cluster configuration on either *MGMT1* or *MGMT2*:

```
$ crm configure property no-quorum-policy=ignore
$ crm configure property stonith-enabled=false
$ crm configure rsc_defaults resource-stickiness=100
```

As with Cloudera Manager Server Pacemaker configuration, this step disables quorum checks, disables STONITH explicitly, and reduces the likelihood of resources being moved between hosts.

5. Create an Open Cluster Framework (OCF) provider on both *MGMT1* and *MGMT2* for Cloudera Manager Agent for use with Pacemaker:

- a. Create an OCF directory for creating OCF resources for Cloudera Manager:

```
$ mkdir -p /usr/lib/ocf/resource.d/cm
```

- b. Create a Cloudera Manager Agent OCF wrapper as a file at `/usr/lib/ocf/resource.d/cm/agent`, with the following content, on both *MGMT1* and *MGMT2*:

```
#!/bin/sh
#####
# CM Agent OCF script
#####
# Initialization:
: ${__OCF_ACTION=$1}
OCF_SUCCESS=0
OCF_ERROR=1
OCF_STOPPED=7
```

```

#####

meta_data() {
    cat <<END
<?xml version="1.0"?>
<!DOCTYPE resource-agent SYSTEM "ra-api-1.dtd">
<resource-agent name="Cloudera Manager Agent" version="1.0">
<version>1.0</version>

<longdesc lang="en">
    This OCF agent handles simple monitoring, start, stop of the Cloudera
    Manager Agent, intended for use with Pacemaker/corosync for failover.
</longdesc>
<shortdesc lang="en">Cloudera Manager Agent OCF script</shortdesc>

<parameters />

<actions>
<action name="start"           timeout="20" />
<action name="stop"           timeout="20" />
<action name="monitor"        timeout="20" interval="10" depth="0"/>
<action name="meta-data"      timeout="5" />
</actions>
</resource-agent>
END
}

#####

agent_usage() {
    cat <<END
    usage: $0 {start|stop|monitor|meta-data}
    Cloudera Manager Agent HA OCF script - used for managing Cloudera Manager Agent and
    managed processes lifecycle for use with Pacemaker.
END
}

agent_start() {
    service cloudera-scm-agent start
    if [ $? = 0 ]; then
        return $OCF_SUCCESS
    fi
    return $OCF_ERROR
}

agent_stop() {
    service cloudera-scm-agent hard_stop_confirmed
    if [ $? = 0 ]; then
        return $OCF_SUCCESS
    fi
    return $OCF_ERROR
}

agent_monitor() {
    # Monitor _MUST!_ differentiate correctly between running
    # (SUCCESS), failed (ERROR) or _cleanly_ stopped (NOT RUNNING).
    # That is THREE states, not just yes/no.
    service cloudera-scm-agent status
    if [ $? = 0 ]; then
        return $OCF_SUCCESS
    fi
    return $OCF_STOPPED
}

case $__OCF_ACTION in
meta-data)    meta_data
               exit $OCF_SUCCESS
               ;;
start)        agent_start;;
stop)         agent_stop;;
monitor)      agent_monitor;;
usage|help)   agent_usage
)

```

```
        exit $OCF_SUCCESS
        ;;
*)
    agent_usage
    exit $OCF_ERR_UNIMPLEMENTED
    ;;
esac
rc=$?
exit $rc
```

c. Run `chmod` on that file to make it executable:

```
$ chmod 770 /usr/lib/ocf/resource.d/cm/agent
```

6. Test the OCF resource script:

```
$ /usr/lib/ocf/resource.d/cm/agent monitor
```

This script should return the current running status of the SCM agent.

7. Add Cloudera Manager Agent as an OCF-managed resource (either on *MGMT1* or *MGMT2*):

```
$ crm configure primitive cloudera-scm-agent ocf:cm:agent
```

8. Verify that the primitive has been picked up by Pacemaker by running the following command:

```
$ crm_mon
```

For example:

```
>crm_mon
Last updated: Tue Jan 27 15:01:35 2015
Last change: Mon Jan 27 14:10:11 2015ls /
Stack: classic openais (with plugin)
Current DC: CMS1 - partition with quorum
Version: 1.1.11-97629de
2 Nodes configured, 2 expected votes
1 Resources configured
Online: [ MGMT1 MGMT2 ]
cloudera-scm-agent (ocf:cm:agent): Started MGMT2
```

Pacemaker starts managing the status of the `cloudera-scm-agent` service on hosts *MGMT1* and *MGMT2*, ensuring that only one instance is running at a time.



Note: Pacemaker expects that all lifecycle actions, such as start and stop, go through Pacemaker; therefore, running direct `service start` or `service stop` commands on one of the hosts breaks that assumption and could cause Pacemaker to start the service on the other host.

Testing Failover with Pacemaker

Test that Pacemaker can move resources by running the following command, which moves the `cloudera-scm-agent` resource to *MGMT2*:

```
$ crm resource move cloudera-scm-agent MGMT2
```

Test the resource move by connecting to a shell on *MGMT2* and verifying that the `cloudera-scm-agent` and the associated Cloudera Management Services processes are now active on that host. It usually takes a few minutes for the new services to come up on the new host.

Database High Availability Configuration

This section contains additional information you can use when configuring databases for high availability.

Database-Specific Mechanisms

- MariaDB:
Configuring MariaDB for high availability requires configuring MariaDB for replication. For more information, see <https://mariadb.com/kb/en/mariadb/setting-up-replication/>.
- MySQL:
Configuring MySQL for high availability requires configuring MySQL for replication. Replication configuration depends on which version of MySQL you are using. For version 5.1, <http://dev.mysql.com/doc/refman/5.1/en/replication-howto.html> provides an introduction.
MySQL GTID-based replication is not supported.
- PostgreSQL:
PostgreSQL has extensive documentation on high availability, especially for versions 9.0 and higher. For information about options available for version 9.1, see <http://www.postgresql.org/docs/9.1/static/high-availability.html>.
- Oracle:
Oracle supports a wide variety of free and paid upgrades to their database technology that support increased availability guarantees, such as their Maximum Availability Architecture (MAA) recommendations. For more information, see <http://www.oracle.com/technetwork/database/features/availability/oracle-database-maa-best-practices-155386.html>.

Disk-Based Mechanisms

DRBD is an open-source Linux-based disk replication mechanism that works at the individual write level to replicate writes on multiple machines. Although not directly supported by major database vendors (at the time of writing of this document), it provides a way to inexpensively configure redundant distributed disk for disk-consistent databases (such as MySQL, PostgreSQL, and Oracle). For information, see <http://drbd.linbit.com>.

TLS and Kerberos Configuration for Cloudera Manager High Availability

This section contains information about configuring Cloudera Manager for high availability when TLS is enabled.

Configuring TLS for Cloudera Manager High Availability

For information about on setting up TLS security with Cloudera Manager, see [t **Configuring TLS Security for Cloudera Manager**](#).

For high availability, keep the following in mind:

- When creating a certificate for the Cloudera Manager Server, make sure that you use the load balancer FQDN (*CMSHostname*, as described in the TLS configuration document referenced above) as your CN.
- Make sure that the same keystore and truststore are placed at the same locations on both *CMS1* and *CMS2* (or share those locations through a shared network mount on both *CMS1* and *CMS2*).
- After completing the steps in the TLS configuration document referenced above, verify SSL/TLS access by accessing the Cloudera Management Admin Console in a web browser, using the load balancer URL:

```
https://[CMSHostname]:[TLS_Port]
```

- For agent configuration of the Cloudera Management Service hosts:
 - If your agents are configured to authenticate the Cloudera Manager Server, Cloudera recommends sharing the filepath (or copying the files manually) for the `verify_cert_file` option as specified in `/etc/cloudera-scm-agent/config.ini` between *MGMT1* and *MGMT2*. If, however, you are using the `verify_cert_dir` mechanism, the `verify_cert_dir` directory contains symlinks when configured correctly with `c_rehash`. Cloudera recommends regenerating that directory based on actual certificate references on both the primary and secondary hosts.

- If your cluster uses TLS authentication of agents to Cloudera Manager (see [Level 3: Configuring TLS Authentication of Agents to the Cloudera Manager Server](#)), share the certificate and keys specified over NFS or by copying them to the same locations on both *MGMT1* and *MGMT2*.



Note: Remember to restart `cloudera-scm-agent` after making changes to these files or configuration options.

Configuring Kerberos Authentication for Cloudera Manager

- If you have configured *CMS1* to store Cloudera Manager Server KDC access credentials at `/etc/cloudera-scm-server`, they must be mirrored on to the same location on *CMS2*.
- Install Kerberos client libraries on both *CMS1* and *CMS2*, and configure the `/etc/krb5.conf` file identically on both machines.
- Make sure that `/etc/krb5.conf` has been configured and distributed on both *MGMT1* and *MGMT2* hosts.
- If you add the Cloudera Management Service to an existing installation using Kerberos authentication (as you do in the [Installing the Primary](#) on page 361 procedure), your Cloudera Management Service could fail to start and displays following message:

mgmt: Configuration Issues

- **C** hostmonitor (test01-ha-common-2): [Role is missing Kerberos keytab.](#)
- **C** reportsmanager (test01-ha-common-2): [Role is missing Kerberos keytab.](#)
- **C** servicemonitor (test01-ha-common-2): [Role is missing Kerberos keytab.](#)
- **C** activitiymonitor (test01-ha-common-2): [Role is missing Kerberos keytab.](#)
- **C** navigatormetaserver (test01-ha-common-2): [Role is missing Kerberos keytab.](#)

Close

This message is expected, and can be corrected by generating Kerberos credentials for these roles using the Cloudera Manager Admin Console. Select **Administration > Kerberos > Credentials > Generate Credentials**.

For additional instructions on configuring Kerberos authentication for Cloudera Manager, see [Configuring Authentication in Cloudera Manager](#).

Backup and Disaster Recovery

This guide describes the Cloudera Manager backup and disaster recovery (BDR) features, which provide an integrated, easy-to-use solution for enabling data protection in the Hadoop platform.



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Backup and Disaster Recovery Overview

Cloudera Manager provides an integrated, easy-to-use management solution for enabling data protection in the Hadoop platform. Cloudera Manager provides rich functionality aimed towards replicating data stored in HDFS and accessed through Hive across data centers for disaster recovery scenarios. When critical data is stored on HDFS, Cloudera Manager provides the necessary capabilities to ensure that the data is available at all times, even in the face of the complete shutdown of a data center.

Cloudera Manager also provides the ability to schedule, save and (if needed) restore snapshots of HDFS directories and HBase tables.

Cloudera Manager provides key capabilities that are fully integrated into the Cloudera Manager Admin Console:

- **Select** - Choose the key datasets that are critical for your business operations.
- **Schedule** - Create an appropriate schedule for data replication or snapshots – trigger replication and snapshots as frequently as is appropriate for your business needs.
- **Monitor** - Track progress of your snapshots and replication jobs through a central console and easily identify issues or files that failed to be transferred.
- **Alert** - Issue alerts when a snapshot or replication job fails or is aborted so that the problem can be diagnosed expeditiously.

Replication capabilities work seamlessly across Hive and HDFS – replication can be setup on files or directories in the case of HDFS and on tables in the case of Hive—without any manual translation of Hive datasets into HDFS datasets or vice-versa. Hive metastore information is also replicated which means that the applications that depend upon the table definitions stored in Hive will work correctly on the replica side as well as the source side as table definitions are updated.

Built on top of a hardened version of `distcp`—the replication uses the scalability and availability of MapReduce and YARN to parallelize the copying of files using a specialized MapReduce job or YARN application that diffs and transfers only changed files from each Mapper to the replica side efficiently and quickly.

Also available is the ability to do a “Dry Run” to verify configuration and understand the cost of the overall operation before actually copying the entire dataset.

Port Requirements

You must ensure that the following ports are open and accessible across clusters to allow communication between the source and destination Cloudera Manager servers and the HDFS, Hive, MapReduce, and YARN hosts:

- Cloudera Manager Admin Console port: Default is 7180.
- HDFS NameNode port: Default is 8020.
- HDFS DataNode port: Default is 50010.
- WebHDFS port: Default is 50070.

See [Ports](#) for more information, including how to verify the current values for these ports.

Data Replication

Cloudera Manager provides rich functionality for replicating data (stored in HDFS or accessed through Hive) across data centers. When critical data is stored on HDFS, Cloudera Manager provides the necessary capabilities to ensure that the data is available at all times, even in the face of the complete shutdown of a data center.

For recommendations on using data replication and Sentry authorization, see [Configuring Sentry to Enable BDR Replication](#).

In Cloudera Manager 5, replication is supported between CDH 5 or CDH 4 clusters. In Cloudera Manager 5, support for HDFS and Hive replication is as follows.



Important: To use HDFS replication, both the target and source HDFS services must use Kerberos authentication, or both must not use Kerberos authentication. See [Enabling Replication Between Clusters in Different Kerberos Realms](#) on page 393.

Supported Replication Scenarios

- **HDFS and Hive**
 - Cloudera Manager 4 with CDH 4 to Cloudera Manager 5 with CDH 4.
 - Cloudera Manager 5 with CDH 4 to Cloudera Manager 4.7.3 or later with CDH 4.
 - Cloudera Manager 5 with CDH 4 to Cloudera Manager 5 with CDH 4.
 - Cloudera Manager 5 with CDH 5 to Cloudera Manager 5 with CDH 5.
 - Cloudera Manager 4 or 5 with CDH 4.4 or later to Cloudera Manager 5 with CDH 5.
 - Cloudera Manager 5 with CDH 5 to Cloudera Manager 5 with CDH 4.4 or later.
 - (HDFS only) Within one Cloudera Manager instance, from one directory to another directory within the same cluster or to a different cluster. Both clusters must be running CDH 4.8 or higher.
- **SSL**
 - Between CDH 5.0 with SSL and CDH 5.0 with SSL.
 - Between CDH 5.0 with SSL and CDH 5.0 without SSL.
 - From a CDH 5.1 source cluster with SSL and YARN.

Unsupported Replication Scenarios

- **HDFS and Hive**
 - Cloudera Manager 5 with CDH 5 as the *source*, and Cloudera Manager 4 with CDH 4 as the *target*.
 - Between Cloudera Enterprise and any Cloudera Manager free edition: Cloudera Express, Cloudera Standard, Cloudera Manager Free Edition.
 - Between CDH 5 and CDH 4 (in either direction) where the replicated data includes a directory that contains a large number of files or subdirectories (several hundreds of thousands of entries), causing out-of-memory errors. This is because of limitations in the WebHDFS API. The workaround is to increase the heap size as follows:
 1. On the target Cloudera Manager instance, go to the HDFS service page.
 2. Click the **Configuration** tab.
 3. Expand the **Service-Wide** category.
 4. Click **Advanced > HDFS Replication Advanced Configuration Snippet**.
 5. Increase the heap size by adding a key-value pair, for instance, `HADOOP_CLIENT_OPTS=-Xmx1g`. In this example, `1g` sets the heap size to 1 GB. This value should be adjusted depending on the number of files and directories being replicated.

- Replication involving HDFS data from CDH 5 HA to CDH 4 clusters or CDH 4 HA to CDH5 clusters will fail if a NameNode failover happens during replication. This is because of limitations in the CDH WebHDFS API.
- **HDFS**
 - Between a source cluster that has encryption over-the-wire enabled and a target cluster running CDH 4.0. This is because the CDH 4 client is used for replication in this case, and it does not support this.
 - From CDH 5 to CDH 4 where there are URL-encoding characters such as % in file and directory names. This is because of a bug in the CDH 4 WebHDFS API.
 - HDFS replication does not work from CDH 5 to CDH 4 with different realms when using older JDK versions. Use JDK 7 or upgrade to JDK6u34 or later on the CDH 4 cluster to avoid this issue.
- **Hive**
 - *With* data replication, between a *source* cluster that has encryption enabled and a *target* cluster running CDH 4. This is because the CDH 4 client used for replication does not support encryption.
 - *Without* data replication, between a *source* cluster running CDH 4 and a *target* cluster that has encryption enabled.
 - Between CDH 4.2 or later and CDH 4, if the Hive schema contains views.
 - With the same cluster as both source and destination
 - Replication from CDH 4 to CDH 5 HA can fail if a NameNode failover happens during replication.
 - Hive replication from CDH 5 to CDH 4 with different realms with older JDK versions, if data replication is enabled (since this involves HDFS replication). Use JDK 7 or upgrade to JDK6u34 or later on the CDH 4 cluster to avoid this issue.
 - Hive replication from CDH 4 to CDH 5 with different realms with older JDK versions (even without data replication enabled). Use JDK 7 or upgrade to JDK6u34 or later on the CDH 4 cluster to avoid this issue.
 - Cloudera Manager 5.2 only supports replication of Impala UDFs if running CDH 5.2 or later. In clusters running Cloudera Manager 5.2 and a CDH version earlier than 5.2 that include Impala User-Defined Functions (UDFs), Hive replication will succeed, but replication of the Impala UDFs will be skipped.



Note: If the `hadoop.proxyuser.hive.groups` configuration has been changed to restrict access to the Hive Metastore Server to certain users/groups only, then the `hdfs` group or a group containing the `hdfs` user must also be included in the list of groups specified in order for Hive replication to work. This can be specified either on the Hive service as an override, or in the core-site HDFS configuration. This note applies to configuration settings on both the source and target clusters.

- **SSL**
 - From a CDH 4.x source cluster with SSL.
 - From CDH 5.0 source cluster with SSL and YARN (because of a YARN bug).
 - Between CDH 5.0 with SSL and CDH 4.x.
- **Kerberos**
 - From a source cluster configured to use Kerberos authentication to a target cluster that is not configured to use Kerberos authentication.
 - From a source cluster not configured to use Kerberos authentication to a target cluster that is configured to use Kerberos authentication.

Designating a Replication Source

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

The Cloudera Manager Server that you are logged in to will be treated as the destination of replications setup via that Cloudera Manager. From the Admin Console of this destination Cloudera Manager, you can designate a peer Cloudera Manager Server which will be treated as a source of HDFS and Hive data for replication.

Configuring a Peer Relationship

1. Go to the **Peers** page by selecting **Administration > Peers**. The Peers page displays. If there are no existing peers, you will see only an **Add Peer** button in addition to a short message. If you have existing peers, they are listed in the Peers list.
2. Click the **Add Peer** button.
3. In the Add Peer pop-up, provide a name, the URL (including the port) of the Cloudera Manager Server that will act as the source for the data to be replicated, and the login credentials for that server.



Important: The role assigned to the login on the source server must be either a *User Administrator* or a *Full Administrator*.

Cloudera recommends that SSL be used and a warning is shown if the URL scheme is `http` instead of `https`. Once both peers have been configured to use SSL/TLS, add the remote source Cloudera Manager's SSL certificate to the local Cloudera Manager truststore, and vice versa.

4. Click the **Add Peer** button in the pop-up to create the peer relationship. The peer is added to the Peers list.
5. To test the connectivity between your Cloudera Manager Server and the peer, select **Actions > Test Connectivity**.

Modifying Peers

1. Go to the **Peers** page by selecting **Administration > Peers**. The Peers page displays. If there are no existing peers, you will see only an **Add Peer** button in addition to a short message. If you have existing peers, they are listed in the Peers list.
2. Choose an action and follow the procedure:
 - **Edit**
 1. From the **Actions** menu for the peer, select **Edit**.
 2. Make your changes.
 3. Click **Update Peer** to save your changes.
 - **Delete** - From the **Actions** menu for the peer, select **Delete**.

Configuring Peers with SAML Authentication

If your cluster uses [SAML Authentication](#) do the following before creating a peer:

1. [Create a Cloudera Manager user account](#) that has the **User Administrator** or **Full Administrator** role.

You can also use an existing user that has one of these roles. Since you will only use this user to create the peer relationship, you can delete the user account after adding the peer.

2. Create or modify the peer, as described in this topic.
3. (Optional) [Delete the Cloudera Manager user account](#) you just created.

HBase Replication

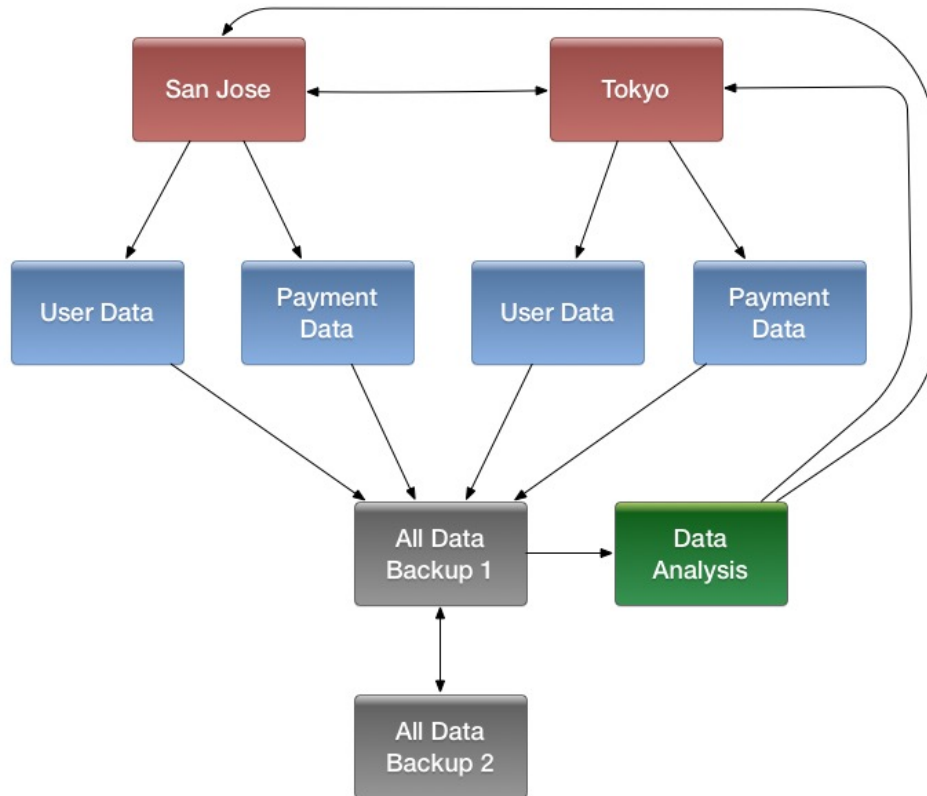
If your data is already in an HBase cluster, replication is useful for getting the data into additional HBase clusters. In HBase, cluster replication refers to keeping one cluster state synchronized with that of another cluster, using the write-ahead log (WAL) of the source cluster to propagate the changes. Replication is enabled at column family granularity. Before enabling replication for a column family, create the table and all column families to be replicated, on the destination cluster.

Cluster replication uses an active-push methodology. An HBase cluster can be a source (also called *active*, meaning that it writes new data), a destination (also called *passive*, meaning that it receives data using replication), or can fulfill both roles at once. Replication is asynchronous, and the goal of replication is consistency.

When data is replicated from one cluster to another, the original source of the data is tracked with a cluster ID, which is part of the metadata. In CDH 5, all clusters that have already consumed the data are also tracked. This prevents replication loops.

Common Replication Topologies

- A central source cluster might propagate changes to multiple destination clusters, for failover or due to geographic distribution.
- A source cluster might push changes to a destination cluster, which might also push its own changes back to the original cluster.
- Many different low-latency clusters might push changes to one centralized cluster for backup or resource-intensive data-analytics jobs. The processed data might then be replicated back to the low-latency clusters.
- Multiple levels of replication can be chained together to suit your needs. The following diagram shows a hypothetical scenario. Use the arrows to follow the data paths.



At the top of the diagram, the `San Jose` and `Tokyo` clusters, shown in red, replicate changes to each other, and each also replicates changes to a `User Data` and a `Payment Data` cluster.

Each cluster in the second row, shown in blue, replicates its changes to the `All Data Backup 1` cluster, shown in grey. The `All Data Backup 1` cluster replicates changes to the `All Data Backup 2` cluster (also shown in grey), as well as the `Data Analysis` cluster (shown in green). `All Data Backup 2` also propagates any of its own changes back to `All Data Backup 1`.

The `Data Analysis` cluster runs MapReduce jobs on its data, and then pushes the processed data back to the `San Jose` and `Tokyo` clusters.

Points to Note about Replication

- The timestamps of the replicated HLog entries are kept intact. In case of a collision (two entries identical as to row key, column family, column qualifier, and timestamp) only the entry arriving later will be read.
- Increment Column Values (ICVs) are treated as simple puts when they are replicated. In the case where each side of replication is active (new data originates from both sources, which then replicate each other), this may be undesirable, creating identical counters that overwrite one another. (See <https://issues.apache.org/jira/browse/HBase-2804>.)
- Make sure the source and destination clusters are time-synchronized with each other. Cloudera recommends you use Network Time Protocol (NTP).

- Some changes are not replicated and must be propagated through other means, such as [Snapshots](#) or [CopyTable](#).
 - Data that existed in the active cluster before replication was enabled.
 - Operations that bypass the WAL, such as when using BulkLoad or API calls such as `writeToWal(false)`.
 - Table schema modifications.

Requirements

Before configuring replication, make sure your environment meets the following requirements:

- You must manage ZooKeeper yourself. It must not be managed by HBase, and must be available throughout the deployment.
- Each host in both clusters must be able to reach every other host, including those in the ZooKeeper cluster.
- Both clusters must be running the same major version of CDH; for example CDH 4 or CDH 5.
- Every table that contains families that are scoped for replication must exist on each cluster and have exactly the same name.
- HBase version 0.92 or greater is required for complex replication topologies, such as active-active.

Deploying HBase Replication

Follow these steps to enable replication from one cluster to another.



Important: To run replication-related HBase commands, your user must have HBase administrator permissions. If ZooKeeper uses Kerberos, [configure HBase Shell to authenticate to ZooKeeper using Kerberos](#) before attempting to run replication-related commands. No replication-related ACLs are available at this time.

1. Configure and start the source and destination clusters. Create tables with the same names and column families on both the source and destination clusters, so that the destination cluster knows where to store data it receives. All hosts in the source and destination clusters should be reachable to each other.
2. On the source cluster, enable replication in Cloudera Manager, or by setting `hbase.replication` to `true` in `hbase-site.xml`.
3. Obtain Kerberos credentials as the HBase principal. Substitute your `fully.qualified.domain.name` and `realm` in the following command:

```
$ kinit -k -t /etc/hbase/conf/hbase.keytab
hbase/fully.qualified.domain.name@YOUR-REALM.COM
```

4. On the source cluster, in HBase Shell, add the destination cluster as a peer, using the `add_peer` command. The syntax is as follows:

```
add_peer 'ID', 'CLUSTER_KEY'
```

The ID must be a short integer. To compose the `CLUSTER_KEY`, use the following template:

```
hbase.zookeeper.quorum:hbase.zookeeper.property.clientPort:zookeeper.znode.parent
```

If both clusters use the same ZooKeeper cluster, you must use a different **zookeeper.znode.parent**, because they cannot write in the same folder.

5. On the source cluster, configure each column family to be replicated by setting its `REPLICATION_SCOPE` to `1`, using commands such as the following in HBase Shell.

```
hbase> disable 'example_table'
hbase> alter 'example_table', {NAME => 'example_family', REPLICATION_SCOPE => '1'}
hbase> enable 'example_table'
```

6. Verify that replication is occurring by examining the logs on the source cluster for messages such as the following.

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 10.10.1.49:62020
```

7. To verify the validity of replicated data, use the included `VerifyReplication` MapReduce job on the source cluster, providing it with the ID of the replication peer and table name to verify. Other options are available, such as a time range or specific families to verify.

The command has the following form:

```
hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication
[--starttime=timestamp] [--stoptime=timestamp] [--families=comma separated list of
families] <peerId> <tablename>
```

The `VerifyReplication` command prints `GOODROWS` and `BADROWS` counters to indicate rows that did and did not replicate correctly.

Guidelines for Replication across Three or More Clusters

When configuring replication among three or more clusters, Cloudera recommends you enable `KEEP_DELETED_CELLS` on column families in the destination cluster, where `REPLICATION_SCOPE=1` in the source cluster. The following commands show how to enable this configuration using HBase Shell.

- On the source cluster:

```
create 't1',{NAME=>'f1', REPLICATION_SCOPE=>1}
```

- On the destination cluster:

```
create 't1',{NAME=>'f1', KEEP_DELETED_CELLS=>'true'}
```

Disabling Replication at the Peer Level

Use the command `disable_peer ("<peerID>")` to disable replication for a specific peer. This will stop replication to the peer, but the logs will be kept for future reference.

To re-enable the peer, use the command `enable_peer (<"peerID">)`. Replication resumes where it was stopped.

Examples:

- To disable peer 1:

```
disable_peer("1")
```

- To re-enable peer 1:

```
enable_peer("1")
```

Stopping Replication in an Emergency

If replication is causing serious problems, you can stop it while the clusters are running.



Warning: Do this only in case of a serious problem; it may cause data loss.

To stop replication in an emergency:

Open the shell on the source cluster and use the `stop_replication` command. For example:

```
hbase(main):001:0> stop_replication
```

Already queued edits will be replicated after you use the `disable_table_replication` command, but new entries will not. See [Understanding How WAL Rolling Affects Replication](#) on page 384.

To start replication again, use the `enable_peer` command.

Initiating Replication When Data Already Exists

You may need to start replication from some point in the past. For example, suppose you have a primary HBase cluster in one location and are setting up a disaster-recovery (DR) cluster in another. To initialize the DR cluster, you need to copy over the existing data from the primary to the DR cluster, so that when you need to switch to the DR cluster you have a full copy of the data generated by the primary cluster. Once that is done, replication of new data can proceed as normal.

To start replication from an earlier point in time, run a `copyTable` command (defining the start and end timestamps), while enabling replication. Proceed as follows:

1. Start replication and note the timestamp.
2. Run the `copyTable` command with an end timestamp equal to the timestamp you noted in the previous step.



Note: Because replication starts from the current WAL, some key values may be copied to the destination cluster by both the replication and the `copyTable` job. This is not a problem because this is an idempotent operation (one that can be applied multiple times without changing the result).

Replicating Pre-existing Data in an Active-Active Deployment

In the case of active-active replication, run the `copyTable` job before starting the replication. (If you start the job after enabling replication, the second cluster will re-send the data to the first cluster, because `copyTable` does not edit the `clusterId` in the mutation objects. Proceed as follows:

1. Run the `copyTable` job and note the start timestamp of the job.
2. Start replication.
3. Run the `copyTable` job again with a start time equal to the start time you noted in step 1.

This results in some data being pushed back and forth between the two clusters; but it minimizes the amount of data.

Understanding How WAL Rolling Affects Replication

When you add a new peer cluster, it only receives new writes from the source cluster **since the last time the WAL was rolled**.

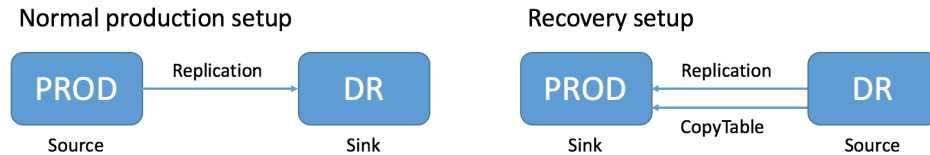
The following diagram shows the consequences of adding and removing peer clusters with unpredictable WAL rolling occurring. Follow the time line and notice which peer clusters receive which writes. Writes that occurred before the WAL is rolled are **not** retroactively replicated to new peers that were not participating in the cluster before the WAL was rolled.

Configuring Secure HBase Replication

If you want to make HBase Replication secure, follow the instructions under [HBase Authentication](#).

Restoring Data From A Replica

One of the main reasons for replications is to be able to restore data, whether during disaster recovery or for other reasons. During restoration, the *source* and *sink* roles are reversed. The source is the replica cluster, and the sink is the cluster that needs restoration. This can be confusing, especially if you are in the middle of a disaster recovery scenario. The following image illustrates the role reversal between normal production and disaster recovery.



Follow these instructions to recover HBase data from a replicated cluster in a disaster recovery scenario.

1. Change the value of the column family property `REPLICATION_SCOPE` on the sink to 0 for each column to be restored, so that its data will not be replicated during the restore operation.
2. Change the value of the column family property `REPLICATION_SCOPE` on the source to 1 for each column to be restored, so that its data will be replicated.
3. Use the `CopyTable` or `distcp` commands to import the data from the backup to the sink cluster, as outlined in [Initiating Replication When Data Already Exists](#) on page 384.
4. Add the sink as a replication peer to the source, using the `add_peer` command as discussed in [Deploying HBase Replication](#) on page 382. If you used `distcp` in the previous step, restart or rolling restart both clusters, so that the RegionServers will pick up the new files. If you used `CopyTable`, you do not need to restart the clusters. New data will be replicated as it is written.
5. When restoration is complete, change the `REPLICATION_SCOPE` values back to their values before initiating the restoration.

Replication Caveats

- Two variables govern replication: `hbase.replication` as described above under [Deploying HBase Replication](#) on page 382, and a replication znode. Stopping replication (using `stop_replication` as above) sets the znode to `false`. Two problems can result:
 - If you add a new RegionServer to the active cluster while replication is stopped, its current log will not be added to the replication queue, because the replication znode is still set to `false`. If you restart replication at this point (using `enable_peer`), entries in the log will not be replicated.
 - Similarly, if a log rolls on an existing RegionServer on the active cluster while replication is stopped, the new log will not be replicated, because the replication znode was set to `false` when the new log was created.
- In the case of a long-running, write-intensive workload, the destination cluster may become unresponsive if its meta-handlers are blocked while performing the replication. CDH 5 provides three properties to deal with this problem:
 - `hbase.regionserver.replication.handler.count` - the number of replication handlers in the destination cluster (default is 3). Replication is now handled by separate handlers in the destination cluster to avoid the above-mentioned sluggishness. Increase it to a high value if the ratio of active to passive RegionServers is high.
 - `replication.sink.client.retries.number` - the number of times the HBase replication client at the sink cluster should retry writing the WAL entries (default is 1).
 - `replication.sink.client.ops.timeout` - the timeout for the HBase replication client at the sink cluster (default is 20 seconds).

- For namespaces, tables, column families, or cells with associated ACLs, the ACLs themselves are not replicated. The ACLs need to be re-created manually on the target table. This behavior opens up the possibility for the ACLs could be different in the source and destination cluster.

HDFS Replication

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

HDFS replication enables you to copy (replicate) your HDFS data from one HDFS service to another, keeping the data set on the *target* service synchronized with the data set on the *source* service, based on a user-specified replication schedule. The target service needs to be managed by the Cloudera Manager Server where the replication is being set up, and the source service could either be managed by that same server or by a peer Cloudera Manager Server.



Important: To use HDFS replication, both the target and source HDFS services must use Kerberos authentication, or both must not use Kerberos authentication. See [Enabling Replication Between Clusters in Different Kerberos Realms](#) on page 393.



Note: If your replication job takes a long time to complete and files change before the replication finishes, the replication may fail. Consider making the directories snapshottable, so that the replication job creates snapshots of the directories before copying the files, and then copies files from these snapshottable directories when executing the replication. See [Using Snapshots with Replication](#) on page 392.

Configuring Replication of HDFS Data

1. Verify that your cluster conforms to the [supported replication scenarios](#).
2. If you are using different Kerberos principals for the source and destination clusters, you will need to add the *destination* principal as a proxy user on the *source* cluster. For example, if you are using the `hdfs_src` principal on the source cluster and the `hdfs_dest` principal on the destination cluster, you must add the following properties to the HDFS service's **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** property on the *source* cluster.

```
<property>
  <name>hadoop.proxyuser.hdfsdest.groups</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hdfsdest.hosts</name>
  <value>*</value>
</property>
```

Deploy client configuration and restart all services on the *source* cluster.

3. If the source cluster is managed by a different Cloudera Manager server from the target cluster, [configure a peer relationship](#).
4. Do one of the following:
 - From the **Backup** tab,
 1. Select **Replications**.
 2. Click **Create** and choose **HDFS Replication**.
 - From the **Clusters** tab:
 1. Go to the HDFS service.
 2. In the HDFS Summary section, select the **Replication** link.
 3. Click **Create** and choose **HDFS Replication**.

The **Create Replication** dialog displays.

5. Click the **Source** field and select the source HDFS service from the HDFS services managed by the peer Cloudera Manager Server or the HDFS services managed by the Cloudera Manager Server whose Admin Console you are logged into.
6. Enter the path to the directory (or file) you want to replicate (the source).
7. Click the Destination field and select the target HDFS service from the HDFS services managed by the Cloudera Manager Server whose Admin Console you are logged into.
8. Enter the path where the target files should be placed.
9. Select a schedule. You can have it run immediately, run once at a scheduled time in the future, or at regularly scheduled intervals. If you select **Once** or **Recurring** you are presented with fields that let you set the date and time and (if appropriate) the interval between runs.
10. If you want to modify the parameters of the job, click **More Options**. Here you can change the following parameters:

- **MapReduce Service** - The MapReduce or YARN service to use.
- **Scheduler Pool** - The scheduler pool to use.
- **Run as** - The user that should run the job. By default this is `hdfs`. If you want to run the job as a different user, you can enter that here. If you are using Kerberos, you *must* provide a user name here, and it must be one with an ID greater than 1000. Verify that the user running the job has a home directory, `/user/<username>`, owned by `username:supergroup` in HDFS.
- **Log path** - An alternative path for the logs.
- **Maximum map slots** and **Maximum bandwidth** - Limits for the number of map slots and for bandwidth per mapper. The defaults are unlimited.
- **Abort on error** - Whether to abort the job on an error (default is not to do so). This means that files copied up to that point will remain on the destination, but no additional files will be copied.
- **Replication Strategy** - Whether file replication tasks should be distributed among the mappers statically or dynamically (the default is static). The static replication strategy distributes file replication tasks among the mappers up front statically, trying to achieve a uniform distribution based on the file sizes. The dynamic replication strategy distributes file replication tasks in small sets to the mappers, and as each mapper is done processing its set of tasks, it dynamically picks up and processes the next unallocated set of tasks.
- **Skip Checksum Checks** - Whether to skip checksum checks (the default is to perform them). If checked, checksum validation will not be performed.
- **Delete policy** - Whether files that were removed on the source should also be deleted from the target directory. This policy also determines the handling of files that exist in the target location but are unrelated to the source. There are three options:
 - **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 will be moved to the trash folder.
 - **Delete permanently** - Uses least amount of space, but should be used with caution.
- **Preserve** - Whether to preserve the block size, replication count, permissions, including ACLs, and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the target file system. The default is to preserve these settings as on the source. When **Permission** is checked, and both the source and target clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When **Extended attributes** is checked, and both the source and target clusters support extended attributes, replication preserves them.



Note: To preserve permissions, you must be running as a superuser on the *destination* cluster. You can use the "Run as" 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 when the replication workflow is aborted.

11. Click **Save Schedule**.

To specify additional replication tasks, select **Create > HDFS Replication**.

A replication task appears in the All Replications list, with relevant information about the source and target locations, the timestamp of the last job, and the next scheduled job (if there is a recurring schedule). A scheduled job will show a calendar icon to the left of the task specification. If the task is scheduled to run once, the calendar icon will disappear after the job has run.

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.

From the **Actions** menu for a replication task, you can:

- Test the replication task without actually transferring data ("Dry Run")
- Edit the task configuration
- Run the task (immediately)
- Delete the task
- Disable or enable the task (if the task is on a recurring schedule). When a task is disabled, instead of the calendar icon you will see a Stopped icon, and the job entry will appear in gray.

Viewing Replication Job Status

- While a job is in progress, the calendar icon turns into spinner, and each stage of the replication task is indicated in the message after the replication specification.
- 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 will *not* be copied. As a result, after the initial job, only a subset of the files may actually be copied, and this will be indicated in the success message.
- If the job fails, a ❌ icon displays.
- For Dry Run jobs, the **Dry Run** action tests the replication flow. By default, up to 1024 replicable source files are tested. The actual number of files tested is equal to 1024 divided by the number of mappers, converted to an integer with a minimum value of 1.
- To view more information about a completed job, click the task row in the Replications list. This displays sub-entries for each past job.
- To view detailed information about a past job, click the entry for that job. This opens another sub-entry that shows:
 - A result message
 - The start and end time of the job.
 - A link to the command details for that replication job.
 - Details about the data that was replicated.
- When viewing a sub-entry, you can dismiss the sub-entry by clicking anywhere in its parent entry, or by clicking the return arrow icon at the top left of the sub-entry area.

Hive Replication

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

Hive replication enables you to copy (replicate) your Hive metastore and data from one cluster to another and keep the Hive metastore and data set on the target cluster synchronized with the source based on a user specified replication schedule. The target cluster needs to be managed by the Cloudera Manager Server where the replication is being set up and the source cluster could either be managed by that same server or by a peer Cloudera Manager Server.



Note: If the `hadoop.proxyuser.hive.groups` configuration has been changed to restrict access to the Hive Metastore Server to certain users/groups only, then the `hdfs` group or a group containing the `hdfs` user must also be included in the list of groups specified in order for Hive replication to work. This can be specified either on the Hive service as an override, or in the core-site HDFS configuration. This note applies to configuration settings on both the source and target clusters.



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](#) on page 392.

**Note:**

If you configured [Synchronizing HDFS ACLs and Sentry Permissions](#) on the target cluster for the directory where HDFS data is copied during Hive replication, the permissions that were copied during replication, are overwritten by the HDFS ACL synchronization and are not preserved.

**Note:**

If you are using Kerberos to secure your clusters, see [Enabling Replication Between Clusters in Different Kerberos Realms](#) on page 393 for details about configuring it.

Hive Tables and DDL Commands

Note the following about 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.

Configuring Replication of Hive Data



Note: In CDH 5.2.0 Hive introduces permanent UDFs. JARs for these UDFs are stored in HDFS at a user defined location. If you are replicating Hive data you should also replicate this directory.

1. Verify that your cluster conforms to the [supported replication scenarios](#).
2. If the source cluster is managed by a different Cloudera Manager server from the target 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 the **Replication** tab.

The Schedules tab of the Replications page displays.


4. Click the **Schedule Hive Replication** link.
5. Select the Hive service from one managed by the local Cloudera Manager Server or from one of the Hive services managed by the peer Cloudera Manager Server to be the source of the replicated data.
6. Leave **Replicate All** checked to replicate all the Hive metastore 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 "|" character. For example: `mydbname1 | mydbname2 | mydbname3`.
 - Regular expressions can be used in either database or table fields. For example:

Regular Expression	Result
<code>[\w].+</code>	Any database/table name
<code>(?!myname\b).+</code>	Any database/table except the one named "myname"
<code>db1 db2</code> <code>[\w_]+</code>	Get all tables of the db1 and db2 databases
<code>db1</code> <code>[\w_]+</code> Click the "+" button and then enter <code>db2</code> <code>[\w_]+</code>	Alternate way to get all tables of the db1 and db2 databases

7. Select the target destination. If there is only one Hive service managed by Cloudera Manager available as a target, then this will be specified as the target. If there are more than one Hive services managed by this Cloudera Manager, select from among them.
8. Select a schedule. You can have it run immediately, run once at a scheduled time in the future, or at regularly scheduled intervals. If you select **Once** or **Recurring** you are presented with fields that let you set the date and time and (if appropriate) the interval between runs.
9. Uncheck the **Replicate HDFS Files** checkbox to skip replicating the associated data files.
10. Uncheck the **Replicate Impala Metadata** checkbox to skip replicating Impala metadata. (This option is checked by default.) See [Impala Metadata Replication](#) on page 392.
11. Use the **More Options** section to specify an export location, modify the parameters of the MapReduce job that will perform the replication, and other options. Here you will be able to select a MapReduce service (if there is more than one in your cluster) and change the following parameters:
 - 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 target Hive metastore. In this example, `user.name` is the process user of the HDFS service on the *destination* cluster. The default HDFS location for this export file can be overridden by specifying a path in the **Export Path** field.

 **Note:** In a Kerberized cluster, the HDFS principal on the *source* cluster should have `read`, `write`, and `execute` access to the **Export Path** directory on the *destination* cluster.

- The **Force Overwrite** option, if checked, forces overwriting data in the target metastore if there are incompatible changes detected. For example, if the target metastore was modified and a new partition was added to a table, this option would force deletion of that partition, overwriting the table with the version found on the source.

 **Important:** If the **Force Overwrite** option is not set and the Hive replication process detects incompatible changes on the source cluster, Hive replication will fail. This situation may arise especially with recurring replications, where the metadata associated with an existing database or table on the source cluster changes over time.

- By default, Hive's HDFS data files (say, `/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 Destination field. For example, if you enter a path such as `/ReplicatedData`, then 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). The user is set in the **Run As** option.
- To specify the user that should run the MapReduce job, use the **Run As** option. By default MapReduce jobs run as `hdfs`. If you want to run the MapReduce job as a different user, you can enter that here. If you are using Kerberos, you *must* provide a user name here, and it must be one with an ID greater than 1000.



Note: If you are using different principals on the source and destination clusters, the user running the MapReduce job should have `read` and `execute` permissions on the Hive warehouse directory on the *source* cluster and `superuser` privileges on the *destination* cluster.

- An alternative path for the logs.
- Limits for the number of map slots and for bandwidth per mapper. The defaults are unlimited.
- Whether to abort the job on an error (default is not to abort the job). Check the checkbox to enable this. This means that files copied up to that point will remain on the destination, but no additional files will be copied.
- Whether the file replication strategy should be static or dynamic (default is static). The static replication strategy distributes file replication tasks among the mappers up front statically, trying to achieve a uniform distribution based on the file sizes. The dynamic replication strategy distributes file replication tasks in small sets to the mappers, and as each mapper is done processing its set of tasks, it dynamically picks up and processes the next unallocated set of tasks.
- Whether to skip checksum checks (default is to perform them).
- Whether files that were removed on the source should also be deleted from the target directory. There are three options: keep deleted files (this is the default), delete the files to the HDFS trash, or delete them permanently.
- 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 target file system. The default is to preserve these settings as on the source.



Note: If you leave the setting to preserve permissions, then you must be running as a superuser. You can use the "Run as" option to ensure that is the case.

- Whether to generate alerts for various state changes in the replication workflow. You can alert on failure, on start, on success, or when the replication workflow is aborted.

12 Click **Save Schedule**.

To specify additional replication tasks, select **Create > Hive Replication**.


A replication task appears in the All Replications list, with relevant information about the source and target locations, the timestamp of the last job, and the next scheduled job (if there is a recurring schedule). A scheduled job will show a calendar icon to the left of the task specification. If the task is scheduled to run once, the calendar icon will disappear after the job has run.

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.

From the **Actions** menu for a replication task, you can:

- Test the replication task without actually transferring data ("Dry Run")
- Edit the task configuration
- Run the task (immediately)
- Delete the task
- Disable or enable the task (if the task is on a recurring schedule). When a task is disabled, instead of the calendar icon you will see a Stopped icon, and the job entry will appear in gray.

Viewing Replication Job Status

- While a job is in progress, the calendar icon turns into spinner, and each stage of the replication task is indicated in the message after the replication specification.
- 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 will *not* be copied. As a result, after the initial job, only a subset of the files may actually be copied, and this will be indicated in the success message.
- If the job fails, a  icon displays.

- For Dry Run jobs, the **Dry Run** action tests the replication flow. By default, up to 1024 replicable source files are tested. The actual number of files tested is equal to 1024 divided by the number of mappers, converted to an integer with a minimum value of 1.
- To view more information about a completed job, click the task row in the Replications list. This displays sub-entries for each past job.
- To view detailed information about a past job, click the entry for that job. This opens another sub-entry that shows:
 - A result message
 - The start and end time of the job.
 - A link to the command details for that replication job.
 - Details about the data that was replicated.
- When viewing a sub-entry, you can dismiss the sub-entry by clicking anywhere in its parent entry, or by clicking the return arrow icon at the top left of the sub-entry area.

Impala Metadata Replication

Impala metadata replication is performed as a part of Hive replication. Impala replication is only supported between two CDH 5 clusters. The Impala and Hive services must be running on both clusters. To enable Impala metadata replication, schedule Hive replication as described in [Configuring Replication of Hive Data](#) on page 389. When performing this procedure, confirm that the **Replicate Impala Metadata** checkbox in the **Create Replication** dialog box is checked.

As long as the above conditions are met, the replication of Impala metadata happens automatically as part of Hive replication. Impala metadata replication is enabled by default.

When you select the **Replicate Impala Metadata** property, it ensures that Impala UDFs (user-defined functions) will be available on the target cluster, just as on the source cluster. As part of replicating the UDFs, the binaries in which they're defined are also replicated.

Using Snapshots with Replication

Some replications, especially those that require a long time to finish, can fail because source files are modified during the replication process. You can prevent such failures by using [Snapshots](#) in conjunction with [Replication](#). This use of snapshots is automatic with CDH versions 5.0 and higher. To take advantage of this, you must enable the relevant directories for snapshots (also called making the directory *snapshottable*).

When the replication job runs, it checks to see whether the specified source directory is snapshottable. Before replicating any files, the replication job creates point-in-time snapshots of these directories and uses them as the source for file copies. This ensures that the replicated data is consistent with the source data as of the start of the replication job. The replication job deletes these snapshots after the replication is complete.

A directory is *snapshottable* because it has been enabled for snapshots, or because a parent directory is enabled for snapshots. Subdirectories of a snapshottable directory are included in the snapshot. To enable an HDFS directory for snapshots (to make it snapshottable), see [Enabling HDFS Snapshots](#) on page 408.

Hive Replication with Snapshots

If you are using [Hive Replication](#), Cloudera recommends that you make the **Hive Warehouse Directory** snapshottable. The Hive Warehouse directory is located in the HDFS file system in the location specified by the `hive.metastore.warehouse.dir` property (the default location is `/user/hive/warehouse`).

If you are using external tables in Hive, also make the directories hosting any external tables not stored in the Hive warehouse directory snapshottable.

Similarly, if you are using Cloudera Impala and are replicating any Impala tables using Hive replication, ensure that the storage locations for the tables and associated databases are also snapshottable. See [Enabling HDFS Snapshots](#) on page 408.

Enabling Replication Between Clusters in Different Kerberos Realms

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

If you want to enable replication between clusters that reside in different Kerberos Realms, there are some additional setup steps you need to perform to ensure that the source and target clusters can communicate.



Note: If either the source or target cluster is running Cloudera Manager 4.6 or later, then both clusters (source and target) must be running 4.6 or later. Cross-realm authentication does not work if one cluster is running Cloudera Manager 4.5.x and one is running Cloudera Manager 4.6 or later.

For HDFS replication:

1. On the hosts in the *target* cluster, ensure that the `krb5.conf` file on each host has the following information:
 - The kdc information for the *source* cluster's Kerberos realm.
 - Domain/host to realm mapping for the *source* cluster NameNode hosts.
2. On the *target* cluster, through Cloudera Manager, add the realm of the *source* cluster to the Trusted Kerberos Realms configuration property.
 - a. Go to the HDFS service.
 - b. Click the **Configuration** tab.
 - c. In the search field type "Trusted Kerberos" to find the **Trusted Kerberos Realms** property.
 - d. Enter the source cluster realm.
 - e. Click **Save Changes** to commit the changes.
3. If your Cloudera Manager is less than 5.0.1, you must restart the JobTracker to enable it to pick up the new Trusted Kerberos Realm settings. Failure to restart the JobTracker prior to the first replication attempt may cause the JobTracker to fail.

For Hive replication:

1. Perform the steps described above on the *target* cluster, including restarting the JobTracker.
2. On the hosts in the *source* cluster, ensure that the `krb5.conf` file on each host has the following information:
 - The kdc information for the *target* cluster's Kerberos realm.
 - Domain/host to realm mapping for the *target* cluster NameNode hosts.
3. On the *source* cluster, through Cloudera Manager, add the realm of the *target* cluster to the Trusted Kerberos Realms configuration property.
 - a. Go to the HDFS service.
 - b. Click the **Configuration** tab.
 - c. In the search field type "Trusted Kerberos" to find the **Trusted Kerberos Realms** property.
 - d. Enter the target cluster realm.
 - e. Click **Save Changes** to commit the changes.
4. It is not necessary to restart any services on the source cluster.

Replication of Encrypted Data

Beginning with CDH 5.3, HDFS supports encryption of data at rest (including data accessed through Hive). This section describes the behavior with respect to encryption during replication, depending on whether or not the source and target are in encryption zones, and the procedure for encrypting data in transit between the source and target clusters.

Encrypting Data in Transit Between Clusters



Note: Regardless of whether HDFS encryption is in use, you must always use SSL/TLS to encrypt data during replication.

A source directory and a destination directory may or may not be in an encryption zone. For more information about HDFS encryption zones, see [HDFS Data At Rest Encryption](#). There are four possible scenarios with respect to whether or not replicated data is encrypted:

- Source and target directory are both in an encryption zone - In this case, the data on the target directory is encrypted.
- Source directory is not encrypted, possibly because the source cluster uses a version of CDH earlier than 5.2 (the first version to support encryption zones) but the target directory is in an encryption zone - In this case, the data on the target directory is encrypted.
- Source directory is in an encryption zone and target directory is not - In this case, the data on the target directory is not encrypted.
- Neither the source nor the target directory are in encryption zones - In this case, the data on the target directory is not encrypted.

Even when the source and target directories are both in encryption zones, the data is decrypted as it is read from the source cluster (using the key for the source encryption zone) and encrypted again when it is written to the target cluster (using the key for the target encryption zone). By default, it is passed over the wire as plain text.

During replication, data travels from the source cluster to the destination cluster using distcp. By default, the data in transit is in plain text. To encrypt data on the wire between the source and target using SSL/TLS:

- Enable SSL/TLS for HDFS clients on both the source and the target clusters. For instructions, see [Configuring SSL for HDFS](#). You may also need to configure trust between the SSL certificates on the source and target.
- Enable SSL/TLS for the two peer Cloudera Manager Servers as described here: [Configuring TLS Encryption Only for Cloudera Manager](#).
- Cloudera recommends you also enable SSL/TLS communication between the Cloudera Manager Server and Agents. See [Configuring TLS Security for Cloudera Manager](#) for instructions.

Snapshots

HBase and HDFS snapshots can be created with Cloudera Manager or by using the command line.

- HBase snapshots allow you to create point-in-time backups of tables without making data copies, and with minimal impact on RegionServers. HBase snapshots are supported for clusters running CDH 4.2 or later.
- HDFS snapshots allow you to create point-in-time backups of directories or the entire filesystem without actually cloning the data. These snapshots appear on the filesystem as read-only directories that can be accessed just like any other ordinary directories. HDFS snapshots are supported for clusters running CDH 5 or later. CDH 4 does not support snapshots for HDFS.

Cloudera Manager Snapshot Policies

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

Cloudera Manager enables the creation of snapshot policies that define the directories or tables to be snapshotted, the intervals at which snapshots should be taken, and the number of snapshots that should be kept for each snapshot interval. For example, you can create a policy that takes both daily and weekly snapshots, and specify that 7 daily snapshots and 5 weekly snapshots should be maintained.



Note: You can improve the reliability of [Data Replication](#) on page 378 by also using snapshots. See [Using Snapshots with Replication](#) on page 392.

Managing Snapshot Policies





Note: An HDFS directory must be enabled for snapshots in order to allow snapshot policies to be created for that directory. To designate a HDFS directory as snapshottable, follow the procedure in [Enabling HDFS Snapshots](#) on page 408.


To create a snapshot policy:


1. Click the **Backup** tab in the top navigation bar and select **Snapshots**.

Existing snapshot policies are shown in a list organized by service. Currently running policies (if any) are shown in the **Running Policies** area.

2. To create a new policy, click **+Create**. If no policies currently exist, click the **Create snapshot policy** link. This displays the Create Snapshot Policy pop-up.
3. Select the service for which you want to create a policy from the pull-down list.
4. Provide a name for the policy and optionally a description.
5. Specify the directories or tables that should be included in the snapshot.
 - For an HDFS service, select the paths of the directories that you want to include in the snapshot. The pull-down list will allow you to select only directories that have been enabled for snapshotting. If no directories have been enabled for snapshotting, a warning is displayed.
Click **+** to add another path, **=** to remove a path.
 - For an HBase service, list the tables you want included in your snapshot. You can use a [Java regular expression](#) to specify a set of tables. An example is `finance.*` which will match all tables with names starting with `finance`.
6. Specify the snapshot schedule. You can schedule snapshots hourly, daily, weekly, monthly, or yearly, or any combination of those. Depending on the frequency you've selected, you can specify the time of day to take the snapshot, the day of the week, day of the month, or month of the year, and the number of snapshots to keep at each interval. Each time unit in the schedule information is shared with the time units of larger granularity. That is, the minute value is shared by all the selected schedules, hour by all the schedules for which hour is applicable, and so on. For example, if you specify that hourly snapshots are taken at the half hour, and daily snapshots taken at the hour 20, the daily snapshot will occur at

Schedule  **Hourly snapshots:** Take snapshots every hour at 30 minutes. Keep 1 hourly snapshots. 

Daily snapshots: Take snapshots at 20:30. Keep 1 daily snapshots. 

- To select an interval, check its box. The description will then display the current schedule and the number of snapshots to retain.
- To edit the schedule (time of day, day of week and so on as relevant), and the number of snapshots to keep, click the edit icon () that appears at the end of the description once you check its box. This opens an area with fields you can edit. When you have made your changes, click the **Close** button at the bottom of this area. Your changes will be reflected in the schedule description.

7. Click **More Options** to specify whether alerts should be generated for various state changes in the snapshot workflow. You can alert on failure, on start, on success, or when the snapshot workflow is aborted.

To edit or delete a snapshot policy:

1. Click the **Backup** tab in the top navigation bar and select **Snapshots**.
2. Click the **Actions** menu shown next to a policy and select **Edit** or **Delete**.


Orphaned Snapshots

When a snapshot policy includes a limit on the number of snapshots to keep, Cloudera Manager checks the total number of stored snapshots each time a new snapshot is added, and automatically deletes the oldest existing snapshot if necessary. When a snapshot policy is edited or deleted, files, directories, or tables that were previously included but have now been removed from the policy may leave "orphaned" snapshots behind that will no longer be deleted automatically because they are no longer associated with a current snapshot policy. Cloudera Manager will never select these snapshots for automatic deletion because selection for deletion only occurs when the policy causes a *new* snapshot containing those files, directories, or tables to be made.

Unwanted snapshots can be deleted manually through the Cloudera Manager interface or by creating a command-line script that uses the HDFS or HBase snapshot commands. Orphaned snapshots may be hard to locate for manual deletion. Snapshot policies are automatically given a prefix `cm-auto` followed by a globally unique identifier (guid). For a specific policy, all its snapshots can be located by searching for those whose names start with the prefix `cm-auto- guid` that is unique to that policy. The prefix is prepended to the names of all snapshots created by that policy.

To avoid orphaned snapshots, delete them before editing or deleting the associated snapshot policy, or make note of the identifying name for the snapshots you want to delete. This prefix is displayed in the summary of the policy in the policy list and appears in the delete dialog box. Making note of the snapshot names, including the associated policy prefix, is necessary because the prefix associated with a policy cannot be determined once the policy has been deleted, and snapshot names do not contain recognizable references to snapshot policies.

Viewing Snapshot History

- To view the history of scheduled snapshot jobs, click a policy. This displays a list of the snapshot jobs, and their status.
- Click a snapshot job to view an expanded status for that job. (Click  to return to the previous view.)
- From the expanded status, click the **details** link to view the details for the command. From here you can view error logs and or click **Download Result Data** to a JSON file named `summary.json` that captures information about the snapshot. For example:

```
{ "createdSnapshotCount" : 1,
  "createdSnapshots" : [ { "creationTime" : null,
    "path" : "/user/oozie",
    "snapshotName" :
"cm-auto-f9299438-a6eb-4f6c-90ac-5e86e5b2e283_HOURLY_2013-11-05_05-25-04",
    "snapshotPath" :
"/user/oozie/.snapshot/cm-auto-f9299438-a6eb-4f6c-90ac-5e86e5b2e283_HOURLY_2013-11-05_05-25-04"
  } ],
  "creationErrorCount" : 0,
  "creationErrors" : [ ],
  "deletedSnapshotCount" : 0,
  "deletedSnapshots" : [ ],
  "deletionErrorCount" : 0,
  "deletionErrors" : [ ],
  "processedPathCount" : 1,
  "processedPaths" : [ "/user/oozie" ],
  "unprocessedPathCount" : 0,
  "unprocessedPaths" : [ ]
}
```

See [Managing HDFS Snapshots](#) on page 407 and [Managing HBase Snapshots](#) on page 396 for more information about managing snapshots.

Managing HBase Snapshots

HBase snapshots can be managed using Cloudera Manager or using the command line, as described in these sections:

- [Using Cloudera Manager](#) on page 396
- [Using the Command Line](#) on page 400

Using Cloudera Manager

For HBase (CDH 4.2 or later or CDH 5) services, a Table Browser tab is available where you can view the HBase tables associated with a service on your cluster. From here you can view the currently saved snapshots for your tables, and delete or restore them as appropriate. From the HBase Table Browser tab you can:

- View the HBase tables that you can snapshot.
- Initiate immediate (unscheduled) snapshots of a table.
- View the list of saved snapshots currently being maintained. These may include one-off immediate snapshots, as well as scheduled policy-based snapshots.
- Delete a saved snapshot.

- Restore from a saved snapshot.
- Restore a table from a saved snapshot to a new table (Restore As).

Browsing HBase Tables

To browse the HBase tables to view snapshot activity:

1. From the **Clusters** tab, select your HBase service.
2. Go to the **Table Browser** tab.

Managing HBase Snapshots

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

To take a snapshot,

1. Click a table.
2. Click **Take Snapshot**.
3. Specify the name of the snapshot, and click **Take Snapshot**.

To delete a snapshot, click  and select **Delete**.

To restore a snapshot, click  and select **Restore**.



Warning: If you use coprocessors, the coprocessor must be available on the destination cluster before restoring the snapshot.

To restore a snapshot to a new table, select **Restore As** from the menu associated with the snapshot, and provide a name for the new table.



Warning: If you "Restore As" to an existing table (that is, specify a table name that already exists) the existing table will be overwritten.

Storing HBase Snapshots on Amazon S3

With Cloudera Manager 5.2 or later and CDH 5.2 or later, HBase snapshots can be stored on the cloud storage service Amazon S3 instead of in HDFS.



Note: When HBase snapshots are stored on, or restored from, Amazon S3, a MapReduce (MRv2) job is created to copy the HBase table data and metadata. For this reason, the YARN service must be running on your Cloudera Manager cluster to use this feature.

To configure HBase to store snapshots on Amazon S3, you must have the following information:

1. The *access key ID* for your Amazon S3 account.
2. The *secret access key* for your Amazon S3 account.
3. The path to the directory in Amazon S3 where you want your HBase snapshots to be stored.

Configuring HBase in Cloudera Manager to Store Snapshots in Amazon S3

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

With the above Amazon S3 information at hand, perform the following steps in Cloudera Manager:

1. Open the HBase service page.
2. Select **Scope > HBASE (Service-Wide)**.
3. Select **Category > Backup**.

4. Type `AWS` in the Search box.
5. Enter your Amazon S3 access key ID in the field **AWS S3 access key ID for remote snapshots**.
6. Enter your Amazon S3 secret access key in the field **AWS S3 secret access key for remote snapshots**.
7. Enter the path to the location in Amazon S3 where your HBase snapshots should be stored in the field **AWS S3 path for remote snapshots**.



Warning: Do not use the Amazon S3 location defined by the path entered in **AWS S3 path for remote snapshots** for any other purpose, or directly add or delete content there. Doing so will risk corrupting the metadata associated with the HBase snapshots stored there. Use this path and Amazon S3 location only through Cloudera Manager, and only for managing HBase snapshots.

8. In a terminal window, log in to your Cloudera Manager cluster at the command line and create a `/user/hbase` directory in HDFS. Change the owner of the directory to `hbase`.

```
Example:  
hdfs dfs -mkdir /user/hbase  
hdfs dfs -chown hbase /user/hbase
```

Configuring the Dynamic Resource Pool Used for Exporting and Importing Snapshots in Amazon S3

Dynamic resource pools are used to control the resources available for MapReduce jobs created for HBase snapshots on Amazon S3. By default, MapReduce jobs run against the default dynamic resource pool. To choose a different dynamic resource pool for HBase snapshots stored on Amazon S3, follow these steps:

1. Open the HBase service page.
2. Select **Scope > HBASE (Service-Wide)**.
3. Select **Category > Backup**.
4. Type `Scheduler` in the Search box.
5. Enter name of a dynamic resource pool in the **Scheduler pool for remote snapshots in AWS S3** property.
6. Click **Save Changes**.

HBase Snapshots on Amazon S3 with Kerberos Enabled

By default, when Kerberos is enabled YARN will not allow MapReduce jobs to be run by the system user `hbase`. If Kerberos is enabled on your cluster, you must perform the following steps:

1. Open the YARN service page in Cloudera Manager.
2. Select **Scope > NodeManager**.
3. Select **Category > Security**.
4. In the **Allowed System Users** property, click the **+** sign and add `hbase` to the list of allowed system users.
5. Click **Save Changes**.
6. Restart the YARN service.

Managing HBase Snapshots on Amazon S3 in Cloudera Manager

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

To take HBase snapshots and store them on Amazon S3, perform the following steps:

1. On the HBase service page in Cloudera Manager, click the **Table Browser** tab.
2. Select a table in the Table Browser. If any recent local or remote snapshots already exist, they will be displayed on the right side.
3. In the dropdown for the selected table, click **Take Snapshot**.
4. Enter a name in the **Snapshot Name** field of the **Take Snapshot** dialog box.
5. If Amazon S3 storage is configured [as described above](#), the **Take Snapshot** dialog box's **Destination** section will show a choice of **Local** or **Remote S3**. Select **Remote S3**.
6. Click **Take Snapshot**.

While the **Take Snapshot** command is being executed, a local copy of the snapshot with a name beginning `cm-tmp` followed by an auto-generated filename is displayed in the Table Browser, but this local copy is deleted as soon as the remote snapshot has been stored in Amazon S3. If the command fails without being completed, the temporary local snapshot may be left behind. This copy can be manually deleted or kept as a valid local snapshot. To store a current snapshot in Amazon S3, either execute the **Take Snapshot** command again, selecting **Remote S3** as the **Destination**, or use the HBase command-line tools to manually export the existing temporary local snapshot to Amazon S3.

Deleting HBase Snapshots from Amazon S3

To delete a snapshot stored in Amazon S3:

1. Select the snapshot in the Table Browser.
2. Click the dropdown arrow for the snapshot.
3. Click **Delete**.

Restoring an HBase Snapshot from Amazon S3

To restore an HBase snapshot that is stored in Amazon S3:

1. Select the table in the Table Browser.
2. Click **Restore Table**.
3. Choose **Remote S3** and select the table to restore.
4. Click **Restore**.

Cloudera Manager will create a local copy of the remote snapshot with a name beginning with `cm-tmp` followed by an auto-generated filename, and will use that local copy to restore the table in HBase. Cloudera Manager will then automatically delete the local copy. If the **Restore** command fails without being completed, the temporary copy may be left behind and will be seen in the Table Browser. In that case, delete the local temporary copy manually and re-execute the **Restore** command to restore the table from Amazon S3.

Restoring an HBase Snapshot from Amazon S3 with a New Name

Restoring an HBase snapshot that is stored in Amazon S3 with a new name is a way of cloning the table without affecting the existing table in HBase. To do this, perform the following steps:

1. Select the table in the Table Browser.
2. Click **Restore Table From Snapshot As**.
3. In the **Restore As** dialog box, enter a new name for the table in the **Restore As** field.
4. Select **Remote S3** and choose the desired snapshot in the list of available Amazon S3 snapshots.

Managing Policies for HBase Snapshots in Amazon S3

You can configure policies to automatically create snapshots of HBase tables on an hourly, daily, weekly, monthly or yearly basis. Snapshot policies for HBase snapshots stored in Amazon S3 are configured using the same procedures as for local HBase snapshots. These procedures are described in [Cloudera Manager Snapshot Policies](#) on page 394. The only additional step to perform for snapshots stored in Amazon S3 is to choose **Remote S3** in the **Destination** section of the policy management dialog boxes.



Note: You can only configure a policy as **Local** or **Remote S3** at the time the policy is created. The setting can not be changed later. If the setting is wrong, create a new policy.

While a snapshot is being made based on a snapshot policy, as with snapshots created manually, a local copy of the snapshot is created, in this case with a name beginning `cm-auto` followed by an auto-generated filename. The temporary copy of the snapshot is displayed in the Table Browser, but this local copy is deleted as soon as the remote snapshot has been stored in Amazon S3. If the snapshot procedure fails without being completed, the temporary local snapshot may be left behind. This copy can be manually deleted or kept as a valid local snapshot. To export the HBase snapshot to Amazon S3, use the HBase command-line tools to manually export the existing temporary local snapshot to Amazon S3.

Using the Command Line

About HBase Snapshots

In previous HBase releases, the only way to a backup or to clone a table was to use `CopyTable` or `ExportTable`, or to copy all the `hfiles` in HDFS after disabling the table. The disadvantages of these methods are:

- `CopyTable` and `ExportTable` can degrade `RegionServer` performance.
- Disabling the table means no reads or writes; this is usually unacceptable.

HBase Snapshots allow you to clone a table without making data copies, and with minimal impact on `RegionServers`. Exporting the table to another cluster should not have any impact on the `RegionServers`.

Use Cases

- Recovery from user or application errors
 - Useful because it may be some time before the database administrator notices the error

**Note:**

The database administrator needs to schedule the intervals at which to take and delete snapshots. Use a script or your preferred management tool for this; it is not built into HBase.

- The database administrator may want to save a snapshot right before a major application upgrade or change.

**Note:**

Snapshots are not primarily used for system upgrade protection because they would not roll back binaries, and would not necessarily be proof against bugs or errors in the system or the upgrade.

- Sub-cases for recovery:
 - Rollback to previous snapshot and merge in reverted data
 - View previous snapshots and selectively merge them into production
- Backup
 - Capture a copy of the database and store it outside HBase for disaster recovery
 - Capture previous versions of data for compliance, regulation, archiving
 - Export from snapshot on live system provides a more consistent view of HBase than `CopyTable` and `ExportTable`
- Audit and/or report view of data at a specific time
 - Capture monthly data for compliance
 - Use for end-of-day/month/quarter reports
- Use for Application testing
 - Test schema or application changes on like production data from snapshot and then throw away
 - For example: take a snapshot; create a new table from the snapshot content (schema plus data); manipulate the new table by changing the schema, adding and removing rows, and so on (the original table, the snapshot, and the new table remain independent of each other)
- Offload work
 - Capture, copy, and restore data to another site
 - Export data to another cluster

Where Snapshots Are Stored

The snapshot metadata is stored in the `.hbase_snapshot` directory under the `hbase` root directory (`/hbase/.hbase-snapshot`). Each snapshot has its own directory that includes all the references to the `hfiles`, logs, and metadata needed to restore the table.

`hfiles` needed by the snapshot are in the traditional

`/hbase/data/<namespace>/<tableName>/<regionName>/<familyName>/` location if the table is still using them; otherwise they will be placed in

`/hbase/.archive/<namespace>/<tableName>/<regionName>/<familyName>/`

Zero-copy Restore and Clone Table

From a snapshot you can create a new table (`clone` operation) or restore the original table. These two operations do not involve data copies; instead a link is created to point to the original `hfiles`.

Changes to a cloned or restored table do not affect the snapshot or (in case of a clone) the original table.

If you want to clone a table to another cluster, you need to export the snapshot to the other cluster and then execute the `clone` operation; see [Exporting a Snapshot to Another Cluster](#).

Reverting to a Previous HBase Version

Snapshots don't affect HBase backward compatibility if they are not used.

If you do use the snapshot capability, backward compatibility is affected as follows:

- If you only take snapshots, you can still go back to a previous HBase version
- If you have used `restore` or `clone`, you cannot go back to a previous version unless the cloned or restored tables have no links (there is no automated way to check; you would need to inspect the file system manually).

Storage Considerations

Since the `hfiles` are immutable, a snapshot consists of reference to the files that are in the table at the moment the snapshot is taken. No copies of the data are made during the snapshot operation, but copies may be made when a compaction or deletion is triggered. In this case, if a snapshot has a reference to the files to be removed, the files are moved to an archive folder, instead of being deleted. This allows the snapshot to be restored in full.

Because no copies are performed, multiple snapshots share the same `hfiles`, but in the worst case scenario, each snapshot could have different set of `hfiles` (tables with lots of updates, and compactions).

Configuring and Enabling Snapshots

Snapshots are on by default; to disable them, set the `hbase.snapshot.enabled` property in `hbase-site.xml` to `false`:

```
<property>
  <name>hbase.snapshot.enabled</name>
  <value>
    false
  </value>
</property>
```

To enable snapshots after you have disabled them, set `hbase.snapshot.enabled` to `true`.



Note:

If you have taken snapshots and then decide to disable snapshots, you must delete the snapshots before restarting the HBase master; the HBase master will not start if snapshots are disabled and snapshots exist.


Snapshots don't affect HBase performance if they are not used.

Backup and Disaster Recovery

Shell Commands

You can manage snapshots by using the HBase shell or the HBaseAdmin Java API.

The following table shows actions you can take from the shell:

Action	Shell command	Comments
Take a snapshot of <code>tableX</code> called <code>snapshotX</code>	<pre>snapshot 'tableX', 'snapshotX'</pre>	<p>Snapshots can be taken while a table is disabled, or while a table is online and serving traffic.</p> <ul style="list-style-type: none"> If a table is disabled (using <code>disable <table></code>), an offline snapshot is taken. This snapshot is managed by the master and fully consistent with the state when the table was disabled. This is the simplest and safest method, but it involves a service interruption because the table must be disabled to take the snapshot. In an online snapshot, the table remains available while the snapshot is taken, and incurs minimal performance degradation of normal read/write loads. This snapshot is managed by the master and run on the RegionServers. The current implementation—simple-flush snapshots—provides no causal consistency guarantees. Despite this shortcoming, it offers the same degree of consistency as <code>CopyTable</code> and is a significant improvement.
Restore snapshot <code>snapshotX</code> (it will replace the source table content)	<pre>restore_snapshot 'snapshotX'</pre>	<p>Restoring a snapshot attempts to replace the current version of a table with another version of the table. To run this command, you must disable the target table. The <code>restore</code> command takes a snapshot of the table (appending a timestamp code), and then clones data into the original data and removes data not in the snapshot. If the operation succeeds, the target table is enabled.</p> <div style="border: 1px solid #f08080; padding: 5px; margin-top: 10px;"> <p> Warning: If you use coprocessors, the coprocessor must be available on the destination cluster before restoring the snapshot.</p> </div>
List all available snapshots	<pre>list_snapshots</pre>	
List all available snapshots starting with <code>'mysnapshot_'</code> (regular expression)	<pre>list_snapshots 'my_snapshot_.*'</pre>	
Remove a snapshot called <code>snapshotX</code>	<pre>delete_snapshot 'snapshotX'</pre>	
Create a new table <code>tableY</code> from a snapshot <code>snapshotX</code>	<pre>clone_snapshot 'snapshotX', 'tableY'</pre>	<p>Cloning a snapshot creates a new read/write table that can serve the data kept at the time of the snapshot. The original table and the cloned table can be modified independently without interfering – new data written to one table will not show up on the other.</p>

Taking a Snapshot Using a Shell Script

With HBase in CDH 5.2 and newer, you can take a snapshot using an operating system shell script, such as a Bash script. This is possible because of HBase Shell's new non-interactive mode, which is described in [Accessing HBase by using the HBase Shell](#). This example Bash script illustrates how to take a snapshot in this way. This script is not production-ready, but is provided as an illustration only.

```
#!/bin/bash
# Take a snapshot of the table passed as an argument
# Usage: snapshot_script.sh table_name
# Names the snapshot in the format snapshot-YYYYMMDD

# Parse the arguments
if [ -z $1 ] || [ $1 == '-h' ]; then
    echo "Usage: $0 <table>"
    echo "    $0 -h"
    exit 1
fi

# Modify to suit your environment
export HBASE_PATH=/home/user/hbase
export DATE=`date +%Y%m%d`
echo "snapshot '$1', 'snapshot-DATE'" | $HBASE_PATH/bin/hbase shell -n
status=$?
if [ $status -ne 0 ]; then
    echo "Snapshot may have failed: $status"
fi
exit $status
```

HBase Shell returns an exit code of 0 on success, but a non-zero exit code only indicates the possibility of failure, rather than definite failure. Therefore, your script should check to see if the snapshot was created before trying again, in the event of a reported failure.

Exporting a Snapshot to Another Cluster

You can export any snapshot from one cluster to another. Exporting the snapshot copies the table's hfiles, logs, and the snapshot metadata, from the source cluster to the destination cluster. Specify the `-copy-from` option to copy from a remote cluster to the local cluster or another remote cluster. If you do not specify the `-copy-from` option, the `hbase.rootdir` in the HBase configuration is used, which means that you are exporting from the current cluster. You must specify the `-copy-to` option, to specify the destination cluster.



Note: Snapshots must be enabled on the destination cluster. See [Configuring and Enabling Snapshots](#) on page 401.



Warning: If you use coprocessors, the coprocessor must be available on the destination cluster before restoring the snapshot.

The `ExportSnapshot` tool executes a MapReduce Job similar to `distcp` to copy files to the other cluster. It works at file-system level, so the HBase cluster can be offline.

Run `ExportSnapshot` as the `hbase` user or the user that owns the files. If the user, group, or permissions need to be different on the destination cluster than the source cluster, be sure the destination directory has the correct permissions.

To copy a snapshot called `MySnapshot` to an HBase cluster `srv2` (`hdfs://srv2:8020/hbase`) using 16 mappers:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
hdfs://srv2:<hdfs_port>/hbase -mappers 16
```

To export the snapshot and change the ownership of the files during the copy:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
hdfs://srv2:<hdfs_port>/hbase -chuser MyUser -chgroup MyGroup -chmod 700 -mappers 16
```

You can also use the Java `-D` option in many tools to specify MapReduce or other configuration properties. For example, the following command copies `MY_SNAPSHOT` to `hdfs://cluster2/hbase` using groups of 10 hfiles per mapper:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot
-Dsnapshot.export.default.map.group=10 -snapshot MY_SNAPSHOT -copy-to
hdfs://cluster2/hbase
```

(The number of mappers is calculated as `TotalNumberOfHFiles/10`.)

To export from one remote cluster to another remote cluster, specify both `-copy-from` and `-copy-to` parameters. You could then reverse the direction to restore the snapshot back to the first remote cluster.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-test -copy-from
hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup
```

To specify a different name for the snapshot on the target cluster, use the `-target` option.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-test -copy-from
hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup -target new-snapshot
```

Restrictions**Warning:**

Do not use merge in combination with snapshots. Merging two regions can cause data loss if snapshots or cloned tables exist for this table.

The merge is likely to corrupt the snapshot and any tables cloned from the snapshot. In addition, if the table has been restored from a snapshot, the merge may also corrupt the table. The snapshot may survive intact if the regions being merged are not in the snapshot, and clones may survive if they do not share files with the original table or snapshot. You can use the `SnapshotInfo` tool (see [Information and Debugging](#) on page 406) to check the status of the snapshot. If the status is `BROKEN`, the snapshot is unusable.

- All the Masters and RegionServers must be running CDH 5.
- If you have [enabled](#) the `AccessController Coprocessor` for HBase, only a global administrator can take, clone, or restore a snapshot, and these actions do not capture the ACL rights. This means that restoring a table preserves the ACL rights of the existing table, while cloning a table creates a new table that has no ACL rights until the administrator adds them.
- Do not take, clone, or restore a snapshot during a rolling restart. Snapshots rely on the RegionServers being up; otherwise the snapshot will fail.



Note: This restriction also applies to rolling upgrade, which can currently be done only via Cloudera Manager.

If you are using HBase Replication and you need to restore a snapshot:**Important:**

Snapshot restore is an emergency tool; you need to disable the table and [table replication](#) to get to an earlier state, and you may lose data in the process.

If you are using [HBase Replication](#), the replicas will be out of sync when you restore a snapshot. If you need to restore a snapshot, proceed as follows:

1. Disable the table that is the restore target, and stop the replication
2. Remove the table from both the master and worker clusters
3. Restore the snapshot on the master cluster
4. Create the table on the worker cluster and use `CopyTable` to initialize it.



Note:

If this is not an emergency (for example, if you know that you have lost just a set of rows such as the rows starting with "xyz"), you can create a clone from the snapshot and create a MapReduce job to copy the data that you've lost.

In this case you don't need to stop replication or disable your main table.

Snapshot Failures

Region moves, splits, and other metadata actions that happen while a snapshot is in progress will probably cause the snapshot to fail; the software detects and rejects corrupted snapshot attempts.

Information and Debugging

You can use the `SnapshotInfo` tool to get information about a snapshot, including status, files, disk usage, and debugging information.

Examples:

Use the `-h` option to print usage instructions for the `SnapshotInfo` utility.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -h
Usage: bin/hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo [options]
where [options] are:
  -h|-help           Show this help and exit.
  -remote-dir        Root directory that contains the snapshots.
  -list-snapshots    List all the available snapshots and exit.
  -snapshot NAME     Snapshot to examine.
  -files             Files and logs list.
  -stats             Files and logs stats.
  -schema            Describe the snapshotted table.
```

Use the `-list-snapshots` option to list all snapshots and exit. This option is new in CDH 5.1.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -list-snapshots
SNAPSHOT | CREATION TIME | TABLE NAME
snapshot-test | 2014-06-24T19:02:54 | test
```

Use the `-remote-dir` option with the `-list-snapshots` option to list snapshots located on a remote system.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -remote-dir
s3n://mybucket/mysnapshot-dir -list-snapshots
SNAPSHOT | CREATION TIME | TABLE NAME
snapshot-test | 2014-05-01 10:30 | myTable
```

Use the `-snapshot` option to print information about a specific snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot
Snapshot Info
-----
Name: test-snapshot
Type: DISABLED
Table: test-table
Version: 0
```

```
Created: 2012-12-30T11:21:21
*****
```

Use the `-snapshot` with the `-stats` options to display additional statistics about a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -stats -snapshot snapshot-test
Snapshot Info
-----
  Name: snapshot-test
  Type: FLUSH
  Table: test
  Format: 0
  Created: 2014-06-24T19:02:54

1 HFiles (0 in archive), total size 1.0k (100.00% 1.0k shared with the source table)
```

Use the `-schema` option with the `-snapshot` option to display the schema of a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -schema -snapshot snapshot-test
Snapshot Info
-----
  Name: snapshot-test
  Type: FLUSH
  Table: test
  Format: 0
  Created: 2014-06-24T19:02:54

Table Descriptor
-----
'test', {NAME => 'cf', DATA_BLOCK_ENCODING => 'FAST_DIFF', BLOOMFILTER => 'ROW',
REPLICATION_SCOPE => '0',
COMPRESSION => 'GZ', VERSIONS => '1', TTL => 'FOREVER', MIN_VERSIONS => '0',
KEEP_DELETED_CELLS => 'false',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
```

Use the `-files` option with the `-snapshot` option to list information about files contained in a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot -files
Snapshot Info
-----
  Name: test-snapshot
  Type: DISABLED
  Table: test-table
  Version: 0
  Created: 2012-12-30T11:21:21

Snapshot Files
-----
  52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/bdf29c39da2a4f2b81889eb4f7b18107
  (archive)
  52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/1e06029d0a2a4a709051b417aec88291
  (archive)
  86.8k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/506f601e14dc4c74a058be5843b99577
  (archive)
  52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/5c7f6916ab724eacbcea218a713941c4
  (archive)
  293.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/aec5e33a6564441d9bd423e31fc93abb
  (archive)
  52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/97782b2fbf0743edaacd8fef06ba51e4
  (archive)

6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the source table)
0 Logs, total size 0.0
```

Managing HDFS Snapshots

This page demonstrates how to manage HDFS Snapshots using either Cloudera Manager or the command line.

Contents:

Managing HDFS Snapshots Using Cloudera Manager

For HDFS (CDH 5 only) services, a File Browser tab is available where you can view the HDFS directories associated with a service on your cluster. From here you can view the currently saved snapshots for your files, and delete or restore them as appropriate. From the HDFS File Browser tab you can:

- Designate HDFS directories to be "snapshottable" so snapshots can be created for those directories.
- Initiate immediate (unscheduled) snapshots of a table.
- View the list of saved snapshots currently being maintained. These may include one-off immediate snapshots, as well as scheduled policy-based snapshots.
- Delete a saved snapshot.
- Restore an HDFS directory or file from a saved snapshot.
- Restore an HDFS directory or file from a saved snapshot to a new directory or file (Restore As)



Note: Cloudera Manager does not support snapshot operations for HDFS paths with encryption-at-rest enabled. This limitation is only for Cloudera Manager, and does not effect CDH command-line tools.

Browsing HDFS Directories

To browse the HDFS directories to view snapshot activity:

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.

As you browse the directory structure of your HDFS, basic information about the directory you have selected is shown at the right (owner, group, and so on).

Enabling HDFS Snapshots

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

HDFS directories must be enabled for snapshots in order for snapshots to be created. You cannot specify a directory as part of a snapshot policy unless it has been enabled for snapshotting.

To enable a HDFS directory for snapshots:

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.
3. Verify the Snapshottable Path and click **Enable Snapshots**.



Note: Once you enable snapshots for a directory, you cannot enable snapshots on any of its subdirectories. Snapshots can be taken only on directories that have snapshots enabled.

To disable snapshots for a directory that has snapshots enabled, use the **Disable Snapshots** from the drop-down menu button at the upper right. If there are existing snapshots of the directory, they must be deleted before snapshots can be disabled.

Managing HDFS Snapshots

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

If a directory has been enabled for snapshots:

- The **Take Snapshot** button is present, enabling an immediate snapshot of the directory.
- Any snapshots that have been taken are listed by the time at which they were taken, along with their names and a menu button.

To take a snapshot, click **Take Snapshot**, specify the name of the snapshot, and click **Take Snapshot**. The snapshot is added to the snapshot list.

To delete a snapshot, click  and select **Delete**.

To restore a snapshot, click  and select **Restore**.

For restoring HDFS data, if a MapReduce or YARN service is present in the cluster, then DistributedCopy (distcp) will be used to restore directories, increasing the speed of restoration. The restore popup for HDFS (under More Options) allows selection of either MapReduce or YARN as the MapReduce service. For files, or if a MapReduce or YARN service is not present, a normal copy will be performed. Use of distcp allows configuration of the following options for the snapshot restoration, similar to what is available when configuring a replication:

- **MapReduce Service** - The MapReduce or YARN service to use.
- **Scheduler Pool** - The scheduler pool to use.
- **Run as** - The user that should run the job. By default this is `hdfs`. If you want to run the job as a different user, you can enter that here. If you are using Kerberos, you *must* provide a user name here, and it must be one with an ID greater than 1000. Verify that the user running the job has a home directory, `/user/<username>`, owned by `username:supergroup` in HDFS.
- **Log path** - An alternative path for the logs.
- **Maximum map slots** and **Maximum bandwidth** - Limits for the number of map slots and for bandwidth per mapper. The defaults are unlimited.
- **Abort on error** - Whether to abort the job on an error (default is not to do so). This means that files copied up to that point will remain on the destination, but no additional files will be copied.
- **Skip Checksum Checks** - Whether to skip checksum checks (the default is to perform them). If checked, checksum validation will not be performed.
- **Delete policy** - Whether files that were removed on the source should also be deleted from the target directory. This policy also determines the handling of files that exist in the target location but are unrelated to the source. There are three options:
 - **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 will be moved to the trash folder.
 - **Delete permanently** - Uses least amount of space, but should be used with caution.
- **Preserve** - Whether to preserve the block size, replication count, permissions, including ACLs, and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the target file system. The default is to preserve these settings as on the source. When **Permission** is checked, and both the source and target clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When **Extended attributes** is checked, and both the source and target clusters support extended attributes, replication preserves them.



Note: To preserve permissions, you must be running as a superuser on the *destination* cluster. You can use the "Run as" option to ensure that is the case.

Managing HDFS Snapshots Using the Command Line

For information about managing snapshots using the command line, see [HDFS Snapshots](#).

Cloudera Manager Administration

Managing the Cloudera Manager Server and Agents

This section covers information on managing the Cloudera Manager Server and Agents that run on each host of the cluster.

Starting, Stopping, and Restarting the Cloudera Manager Server

To start the Cloudera Manager Server:

```
$ sudo service cloudera-scm-server start
```

You can stop (for example, to perform maintenance on its host) or restart the Cloudera Manager Server without affecting the other services running on your cluster. Statistics data used by activity monitoring and service monitoring will continue to be collected during the time the server is down.

To stop the Cloudera Manager Server:

```
$ sudo service cloudera-scm-server stop
```

To restart the Cloudera Manager Server:

```
$ sudo service cloudera-scm-server restart
```

Configuring Cloudera Manager Server Ports

Minimum Required Role: [Full Administrator](#)

1. Select **Administration > Settings**.
2. Under the **Ports and Addresses** category, set the following options as described below:

Setting	Description
HTTP Port for Admin Console	Specify the HTTP port to use to access the Server using the Admin Console.
HTTPS Port for Admin Console	Specify the HTTPS port to use to access the Server using the Admin Console.
Agent Port to connect to Server	Specify the port for Agents to use to connect to the Server.

3. Click **Save Changes**.
4. [Restart the Cloudera Manager Server](#).

Moving the Cloudera Manager Server to a New Host

You can move the Cloudera Manager Server if either the Cloudera Manager database server or a current [back up](#) of the Cloudera Manager database is available. To move Cloudera Manager Server:

1. Identify a new host on which to install Cloudera Manager.
2. Install Cloudera Manager on a new host, using the method described under [Install the Cloudera Manager Server Packages](#). Do not install the other components, such as CDH and databases.
3. Copy the entire content of `/var/lib/cloudera-scm-server/` on the old host to that same path on the new host, in the same path. Ensure you preserve permissions and all file content.

4. If the database server is not available:
 - a. Install the database packages on the host that will host the restored database. This could be the same host on which you have just installed Cloudera Manager or it could be a different host. If you used the embedded PostgreSQL database, install the PostgreSQL package as described in [Embedded PostgreSQL Database](#). If you used an external MySQL, PostgreSQL, or Oracle database, reinstall following the instructions in [Cloudera Manager and Managed Service Data Stores](#).
 - b. Restore the backed up databases to the new database installation.
5. Update `/etc/cloudera-scm-server/db.properties` with the database name, database instance name, user name, and password.
6. In `/etc/cloudera-scm-agent/config.ini` on each host, update the `server_host` property to the new hostname and restart the Agents.
7. Start the Cloudera Manager Server. Cloudera Manager should resume functioning as it did before the failure. Because you restored the database from the backup, the server should accept the running state of the Agents, meaning it will not terminate any running processes.

The process is similar with secure clusters, though files in `/etc/cloudera-scm-server` must be restored in addition to the database. See [Cloudera Security](#).

Starting, Stopping, and Restarting Cloudera Manager Agents

Starting Agents

To start Agents, the `supervisord` process, and *all managed service processes*, use one of the following commands:

- **Start**

```
sudo service cloudera-scm-agent start
```

-

Stopping and Restarting Agents

To stop or restart Agents *while leaving the managed processes running*, use one of the following commands:

- **Stop**

```
$ sudo service cloudera-scm-agent stop
```

- **Restart**

```
$ sudo service cloudera-scm-agent restart
```

Hard Stopping and Restarting Agents



Warning: The `hard_stop` and `hard_restart` commands kill all running managed service processes on the host(s) where the command is run.

To stop or restart Agents, the `supervisord` process, and *all managed service processes*, use one of the following commands:

- **Hard Stop**

- RHEL-compatible 7 and higher:

```
$ sudo service cloudera-scm-agent next_stop_hard
$ sudo service cloudera-scm-agent stop
```

- All other Linux distributions:

```
$ sudo service cloudera-scm-agent hard_stop
```

- **Hard Restart**

- RHEL-compatible 7 and higher:

```
$ sudo service cloudera-scm-agent next_stop_hard  
$ sudo service cloudera-scm-agent restart
```

- All other Linux distributions:

```
$ sudo service cloudera-scm-agent hard_restart
```

Hard restart is useful for the following situations:

1. You're upgrading Cloudera Manager and the `supervisord` code has changed between your current version and the new one. To properly do this upgrade you'll need to restart supervisor too.
2. `supervisord` is hung and needs to be restarted.
3. You want to clear out all running state pertaining to Cloudera Manager and managed services.

-

Checking Agent Status

To check the status of the Agent process, use the command:

```
$ sudo service cloudera-scm-agent status
```

Configuring Cloudera Manager Agents

Minimum Required Role: [Full Administrator](#)

Cloudera Manager Agents can be configured globally using properties you set in the Cloudera Manager Admin Console and by setting properties in Agent configuration files.

Configuring Agent Heartbeat and Health Status Options

You can configure the Cloudera Manager Agent heartbeat interval and timeouts to trigger changes in Agent [health](#) as follows:

1. Select **Administration > Settings**.
2. Under the **Performance** category, set the following option:

Property	Description
Send Agent Heartbeat Every	The interval in seconds between each heartbeat that is sent from Cloudera Manager Agents to the Cloudera Manager Server. Default: 15 sec.

3. Under the **Monitoring** category, set the following options:

Property	Description
Set health status to Concerning if the Agent heartbeats fail	The number of missed consecutive heartbeats after which a Concerning health status is assigned to that Agent. Default: 5.

Property	Description
Set health status to Bad if the Agent heartbeats fail	The number of missed consecutive heartbeats after which a Bad health status is assigned to that Agent. Default: 10.

4. Click **Save Changes**.

Configuring the Host Parcel Directory



Important: If you modify the parcel directory location, make sure that all hosts use the same location. Using different locations on different hosts can cause unexpected problems.

To configure the location of distributed parcels:


1. Click **Hosts** in the top navigation bar.
2. Click the **Configuration** tab.
3. Select **Category Parcels**.
4. Configure the value of the **Parcel Directory** property. The setting of the `parcel_dir` property in the [Cloudera Manager Agent configuration file](#) overrides this setting.
5. Click **Save Changes** to commit the changes.
6. [Restart](#) the Cloudera Manager Agent on all hosts.

Agent Configuration File

The Cloudera Manager Agent supports different types of configuration options in the `/etc/cloudera-scm-agent/config.ini` file. You must update the configuration on each host. After changing a property, restart the Agent:

```
$ sudo service cloudera-scm-agent restart
```

Section	Property	Description
[General]	<code>server_host</code> , <code>server_port</code> , <code>listening_port</code> , <code>listening_hostname</code> , <code>listening_ip</code>	<p>Hostname and ports of the Cloudera Manager Server and Agent and IP address of the Agent. Also see Configuring Cloudera Manager Server Ports on page 410 and Ports Used by Cloudera Manager and Cloudera Navigator.</p> <p>The Cloudera Manager Agent configures its hostname automatically. You can also manually specify the hostname the Cloudera Manager Agent uses by updating the <code>listening_hostname</code> property. To manually specify the IP address the Cloudera Manager Agent uses, update the <code>listening_ip</code> property in the same file.</p> <p>To have a CNAME used throughout instead of the regular hostname, an Agent can be configured to use <code>listening_hostname=CNAME</code>. In this case, the CNAME should resolve to the same IP address as the IP address of the hostname on that machine. Users doing this will find that the host inspector will report problems, but the CNAME will be used in all configurations where that's appropriate. This practice is particularly useful for users who would like clients to use <code>namenode.mycluster.company.com</code> instead of <code>machine1234.mycluster.company.com</code>. In this case, <code>namenode.mycluster</code> would be a CNAME for</p>

Section	Property	Description
		machine1234.mycluster, and the generated client configurations (and internal configurations as well) would use the CNAME.
	lib_dir	Directory to store Cloudera Manager Agent state that persists across instances of the agent process and system reboots. The Agent UUID is stored here. Default: /var/lib/cloudera-scm-agent.
	local_filesystem_whitelist	The list of local filesystems that should always be monitored. Default: ext2,ext3,ext4.
	log_file	The path to the Agent log file. If the Agent is being started using the init.d script, /var/log/cloudera-scm-agent/cloudera-scm-agent.out will also have a small amount of output (from before logging is initialized). Default: /var/log/cloudera-scm-agent/cloudera-scm-agent.log.
	max_collection_wait_seconds	Maximum time to wait for all metric collectors to finish collecting data. Default: 10 sec.
	metrics_url_timeout_seconds	Maximum time to wait when connecting to a local role's web server to fetch metrics. Default: 30 sec.
	parcel_dir	Directory to store unpacked parcels. Default: /opt/cloudera/parcels. <div style="border: 1px solid orange; padding: 5px; margin: 10px 0;"> <p> Important: If you modify the parcel directory location, make sure that all hosts use the same location. Using different locations on different hosts can cause unexpected problems.</p> </div> <p>If you want to change this, Cloudera recommends following the procedure documented in Changing the Parcel Directory to change this for all hosts, rather than setting it in each host config.ini file.</p> <p>This property overrides the setting in Cloudera Manager. To use the recommended procedure, you must make sure that this property is commented out in each host config.ini file.</p>
	supervisord_port	The supervisord port. A change takes effect the next time supervisord is restarted (not when the Agent is restarted). Default: 19001.
	task_metrics_timeout_seconds	Maximum time to wait when connecting to a local TaskTracker to fetch task attempt data. Default: 5 sec.

Section	Property	Description
[Security]	use_tls, verify_cert_file, client_key_file, client_keypw_file, client_cert_file	Security-related configuration. See <ul style="list-style-type: none"> Level 3: Configuring TLS Authentication of Agents to the Cloudera Manager Server Level 2: Configuring TLS Verification of Cloudera Manager Server by the Agents Specifying the Cloudera Manager Server Certificate Adding a Host to the Cluster on page 54
[Cloudera]	mgmt_home	Directory to store Cloudera Management Service files. Default: /usr/share/cmfs.
[JDBC]	cloudera_mysql_connector_jar, cloudera_oracle_connector_jar, cloudera_postgresql_jdbc_jar	Location of JDBC drivers. See Cloudera Manager and Managed Service Data Stores . Default: <ul style="list-style-type: none"> MySQL - /usr/share/java/mysql-connector-java.jar Oracle - /usr/share/java/oracle-connector-java.jar PostgreSQL - /usr/share/cmfs/lib/postgresql-version-build.jdbc4.jar

Managing Cloudera Manager Server and Agent Logs

Viewing Logs

To help you troubleshoot problems, you can view the Cloudera Manager Server and Agent logs. You can view these logs in the Logs page or in specific pages for the logs.

Viewing Cloudera Manager Server and Agent Logs in the Logs Page

1. Select **Diagnostics > Logs** on the top navigation bar.
2. Click **Select Sources** to display the log source list.
3. Uncheck the **All Sources** checkbox.
4. Check the **Cloudera Manager** checkbox to view both Agent and Server logs, or click ► to the left of Cloudera Manager, and check either the **Agent** or **Server** checkbox.
5. Click **Search**.

For more information about the Logs page, see [Logs](#).

Viewing the Cloudera Manager Server Log

1. Select **Diagnostics > Server Log** on the top navigation bar.



Note: You can also view the Cloudera Manager Server log at /var/log/cloudera-scm-server/cloudera-scm-server.log on the Server host.

Viewing the Cloudera Manager Agent Log

1. Click the **Hosts** tab.
2. Click the link for the host where you want to see the Agent log.
3. In the **Details** panel, click the **Details** link in the **Host Agent** field.
4. Click the **Agent Log** link.



Note: You can also view the Cloudera Manager Agent log at `/var/log/cloudera-scm-agent/cloudera-scm-agent.log` on the Agent hosts.

Configuring Log Locations

Setting the Cloudera Manager Server Log Location

By default the Cloudera Manager Server log is stored in `/var/log/cloudera-scm-server/`. If there is not enough space in that directory, you can change the location of the parent of the log directory:

1. Stop the Cloudera Manager Server:

```
$ sudo service cloudera-scm-server stop
```

2. Set the `CMF_VAR` environment variable in `/etc/default/cloudera-scm-server` to the new parent directory:

```
export CMF_VAR=/opt
```

3. Create `log/cloudera-scm_server` and `run` directories in the new parent directory and set the owner and group of all directories to `cloudera-scm`. For example, if the new parent directory is `/opt/`, do the following:

```
$ sudo su
$ cd /opt
$ mkdir log
$ chown cloudera-scm:cloudera-scm log
$ mkdir /opt/log/cloudera-scm-server
$ chown cloudera-scm:cloudera-scm log/cloudera-scm-server
$ mkdir run
$ chown cloudera-scm:cloudera-scm run
```

4. Restart the Cloudera Manager Server:

```
$ sudo service cloudera-scm-server start
```

Setting the Cloudera Manager Agent Log Location

By default the Cloudera Manager Agent log is stored in `/var/log/cloudera-scm-agent/`. If there is not enough space in that directory, you can change the location of the log file:

1. Set the `log_file` property in the Cloudera Manager Agent [configuration file](#):

```
log_file=/opt/log/cloudera-scm-agent/cloudera-scm-agent.log
```

2. Create `log/cloudera-scm_agent` directories and set the owner and group to `cloudera-scm`. For example, if the log is stored in `/opt/log/cloudera-scm-agent`, do the following:

```
$ sudo su
$ cd /opt
$ mkdir log
$ chown cloudera-scm:cloudera-scm log
$ mkdir /opt/log/cloudera-scm-agent
$ chown cloudera-scm:cloudera-scm log/cloudera-scm-agent
```

3. Restart the Agent:

```
$ sudo service cloudera-scm-agent restart
```

Changing Hostnames

Minimum Required Role: [Full Administrator](#)



Important: The process described here requires Cloudera Manager and cluster downtime.

After you have installed Cloudera Manager and created a cluster, you may need to update the names of the hosts running the Cloudera Manager Server or cluster services. To update a deployment with new hostnames, follow these steps:

1. Verify if SSL/TLS certificates have been issued for any of the services and make sure to create new SSL/TLS certificates in advance for services protected by SSL/TLS. See [Encryption](#).
2. [Export](#) the Cloudera Manager configuration using one of the following methods:

- Open a browser and go to this URL `http://cm_hostname:7180/api/api_version/cm/deployment`. Save the displayed configuration.
- From terminal type:

```
$ curl -u admin:admin http://cm_hostname:7180/api/api_version/cm/deployment >
cme-cm-export.json
```

If Cloudera Manager SSL is in use, specify the `-k` switch:

```
$ curl -k -u admin:admin http://cm_hostname:7180/api/api_version/cm/deployment >
cme-cm-export.json
```

where `cm_hostname` is the name of the Cloudera Manager host and `api_version` is the correct [version](#) of the API for the version of Cloudera Manager you are using. For example, `http://tcdn5-1.ent.cloudera.com:7180/api/v10/cm/deployment`.

3. [Stop all services](#) on the cluster.
4. [Stop the Cloudera Management Service](#).
5. [Stop the Cloudera Manager Server](#).
6. [Stop the Cloudera Manager Agents](#) on the hosts that will be having the hostname changed.
7. [Back up the Cloudera Manager Server database](#) using `mysqldump`, `pg_dump`, or another preferred backup utility. Store the backup in a safe location.
8. Update names and principals:
 - a. Update the target hosts using standard per-OS/name service methods (`/etc/hosts`, `dns`, `/etc/sysconfig/network`, `hostname`, and so on). Ensure that you remove the old hostname.
 - b. If you are changing the hostname of the host running Cloudera Manager Server do the following:
 - a. Change the hostname per [step 8.a](#).
 - b. Update the Cloudera Manager hostname in `/etc/cloudera-scm-agent/config.ini` on all Agents.
 - c. If the cluster is configured for Kerberos security, do the following:
 - a. Remove old hostname cluster service principals from the KDC database using one of the following:
 - Use the `delprinc` command within `kadmin.local` interactive shell.
 - From the command line:

```
kadmin.local -q "listprincs" | grep -E
"(HTTP|hbase|hdfs|hive|httpfs|hue|impala|mapred|solr|oozie|yarn|zookeeper)[^/]*/[^/]*@"
> cluster-princ.txt
```

Open `cluster-princ.txt` and remove any non-cluster service principal entries within it. Make sure that the default `krbtgt` and other principals you created, or were created by Kerberos by default, are not removed by running the following: `for i in `cat cluster-princ.txt`; do yes yes | kadmin.local -q "delprinc $i"; done.`

- b. Start the Cloudera Manager database and Cloudera Manager Server.

- c. Start the Cloudera Manager Agents on the newly renamed hosts. The Agents should show a current heartbeat in Cloudera Manager.
 - d. Within the Cloudera Manager Admin Console click the **Hosts** tab.
 - e. Select the checkbox next to the host with the new name.
 - f. Select **Actions > Regenerate Keytab**.
9. If one of the hosts that was renamed has a NameNode configured with high availability and automatic failover enabled, reconfigure the ZooKeeper Failover Controller znodes to reflect the new hostname.
- a. Start ZooKeeper Servers.



Warning: All other services, and most importantly HDFS, and the ZooKeeper Failover Controller (FC) role within the HDFS, should not be running.

- b. On one of the hosts that has a ZooKeeper Server role, run `zookeeper-client`.
 - a. If the cluster is configured for Kerberos security, configure ZooKeeper authorization as follows:
 - a. Go to the HDFS service.
 - b. Click the **Instances** tab.
 - c. Click the **Failover Controller** role.
 - d. Click the **Process** tab.
 - e. In the Configuration Files column of the `hdfs/hdfs.sh ["zkfc"]` program, expand **Show**.
 - f. Inspect `core-site.xml` in the displayed list of files and determine the value of the `ha.zookeeper.auth` property, which will be something like:
`digest:hdfs-fcs:TEbW2bgoODa96rO3ZTn7ND5fSOGx0h`. The part after `digest:hdfs-fcs:` is the password (in the example it is `TEbW2bgoODa96rO3ZTn7ND5fSOGx0h`)
 - g. Run the `addauth` command with the password:

```
addauth digest hdfs-fcs:TEbW2bgoODa96rO3ZTn7ND5fSOGx0h
```

- b. Verify that the HA znode exists: `ls /hadoop-ha`.
 - c. Delete the HDFS znode: `rmdir /hadoop-ha/nameservice1`.
 - d. If you *are not* running JobTracker in a high availability configuration, delete the HA znode: `rmdir /hadoop-ha`.
- c. In the Cloudera Manager Admin Console, go to the HDFS service.
 - d. Click the **Instances** tab.
 - e. Select **Actions > Initialize High Availability State in ZooKeeper....**
10. Update the Hive metastore:
- a. Back up the Hive metastore database.
 - b. Go the Hive service.
 - c. Select **Actions > Update Hive Metastore NameNodes** and confirm the command.
11. Update the **Database Hostname** property for each of the cluster roles for which a database is located on the host being renamed. This is required for both Cloudera Management Service roles (Reports Manager, Activity Monitor, Navigator Audit and Metadata Server) and for cluster services such as Hue, Hive, and so on.
12. Start all cluster services.
13. Start the Cloudera Management Service.
14. Deploy client configurations.

Configuring Network Settings

Minimum Required Role: [Full Administrator](#)

To configure a proxy server thorough which data is downloaded to and uploaded from the Cloudera Manager Server, do the following:

1. Select **Administration > Settings**.
2. Click the **Network** category.
3. Configure proxy properties.
4. Click **Save Changes** to commit the changes.


Managing Alerts

Minimum Required Role: [Full Administrator](#)

The **Administration > Alerts** page provides a summary of the settings for alerts in your clusters.

Alert Type The left column lets you select by alert type (Health, Log, or Activity) and within that by service instance. In the case of Health alerts, you can look at alerts for Hosts as well. You can select an individual service to see just the alert settings for that service.

Health/Log/Activity Alert Settings Depending on your selection in the left column, the right hand column show you the list of alerts that are enabled or disabled for the selected service type.

To change the alert settings for a service, click the  next to the service name. This will take you to the Monitoring section of the Configuration tab for the service. From here you can enable or disable alerts and configure thresholds as needed.

Recipients You can also view the list of recipients configured for the enabled alerts.

Configuring Alert Delivery

When you install Cloudera Manager you can configure the mail server you will use with the Alert Publisher. However, if you need to change these settings, you can do so under the Alert Publisher section of the Management Services configuration tab. Under the Alert Publisher role of the Cloudera Manager Management Service, you can configure email or SNMP delivery of alert notifications.


Configuring Alert Email Delivery

Minimum Required Role: [Full Administrator](#)

Sending A Test Alert E-mail

Select the **Administration > Alerts** tab and click the **Send Test Alert** link.

Configuring the List Of Alert Recipient Email Addresses

1. Do one of the following:
 - Select the **Administration > Alerts** tab and click the  to the right of **Recipient(s)**.
 - 1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
 2. Click the **Configuration** tab.
2. Select **Scope > Alert Publisher**.
3. Select **Category > Main**.
4. Locate the **Alerts: Mail Message Recipients** property or search for it by typing its name in the Search box.
5. Configure the **Alerts: Mail Message Recipients** property.
6. Click the **Save Changes** button at the top of the page to save your settings.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

7. Restart the Alert Publisher role.

Configuring Alert Email Properties

1. [Display the Cloudera Management Service](#) status page.
2. Click the **Configuration** tab.
3. Select **Scope > Alert Publisher**.
4. Select **Category > Main** to see the list of properties. To receive email alerts, you must set (or verify) the following settings:
 - Enable email alerts
 - Email protocol to use.
 - Your mail server hostname and port.
 - The username and password of the email user that will be logged into the mail server as the "sender" of the alert emails.
 - A comma-separated list of email addresses that will be the recipients of alert emails.
 - The format of the email alert message. Select **json** if you need the message to be parsed by a script or program.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click the **Save Changes** button at the top of the page to save your settings.
6. Restart the Alert Publisher role.

Configuring Alert SNMP Delivery

Minimum Required Role: [Full Administrator](#)



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Enabling, Configuring, and Disabling SNMP Traps

1. Before you enable SNMP traps, configure the trap receiver (Network Management System or SNMP server) with the Cloudera MIB.
2. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
3. Click the **Configuration** tab.
4. Select **Scope > Alert Publisher > SNMP**.
5. Select **Category > SNMP**
 - Enter the DNS name or IP address of the Network Management System (SNMP server) acting as the trap receiver in the **SNMP NMS Hostname** property.
 - In the **SNMP Security Level** property, select the version of SNMP you are using: SNMPv2, SNMPv3 without authentication and without privacy (`noAuthNoPriv`), or SNMPv3 with authentication and without privacy (`authNoPriv`) and specify the required properties:
 - SNMPv2 - SNMPv2 Community String.
 - SNMPv3 without authentication (`noAuthNoPriv`) - SNMP Server Engine Id and SNMP Security UserName.

- SNMPv3 with authentication (`authNoPriv`) - SNMP Server Engine Id, SNMP Security UserName, SNMP Authentication Protocol, and SNMP Authentication Protocol Pass Phrase.

- You can also change other settings such as the port, retry, or timeout values.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

6. Click **Save Changes** when you are done.
7. Restart the Alert Publisher role.

To disable SNMP traps, remove the hostname from the **SNMP NMS Hostname** property (`alert.snmp.server.hostname`).

Viewing the Cloudera MIB

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Alert Publisher > SNMP**.
4. Select **Category > SNMP**.
5. Locate the **SNMP NMS Hostname** property and click the ? icon to display the property description.
6. Click the **SNMP Mib** link.

Managing Licenses

Minimum Required Role: [Full Administrator](#)

When you install Cloudera Manager, you can select among the following editions: Cloudera Express (no license required), a 60-day Cloudera Enterprise Data Hub Edition trial license, or Cloudera Enterprise (which requires a license). To obtain a Cloudera Enterprise license, fill in this [form](#) or call 866-843-7207.

A Cloudera Enterprise license is required for the following features:

- LDAP and SAML authentication
- Configuration history
- Alerts delivered as SNMP traps
- Backup and disaster recovery
- Operational reports
- Cloudera Navigator
- Commands such as Rolling Restart, History and Rollback, and Send Diagnostic Data

For details see [Cloudera Express and Cloudera Enterprise Features](#).

Accessing the License Page

To access the license page, select **Administration > License**.

If you have a license installed, the license page indicates its status (for example, whether your license is currently valid) and displays the license details: the license owner, the license key, and the expiration date of the license, if there is one.

At the right side of the page a table shows the usage of licensed components based on the number of hosts with those products installed. You can move the cursor over the



to see an explanation of each item.

- **Basic Edition** - a cluster running core CDH services: HDFS, Hive, Hue, MapReduce, Oozie, Sqoop, YARN, and ZooKeeper.

- **Flex Edition** - a cluster running core CDH services plus one of the following: Accumulo, HBase, Impala, Navigator, Solr, Spark.
- **Data Hub Edition** - a cluster running core CDH services plus any of the following: Accumulo, HBase, Impala, Navigator, Solr, Spark.

License Expiration

When a Cloudera Enterprise license expires, the following occurs:

- Cloudera Enterprise Data Hub Edition Trial - Enterprise features are no longer available.
- Cloudera Enterprise - Cloudera Manager Admin Console displays a banner indicating license expiration. Contact Cloudera Support to receive an updated license. In the meanwhile, all enterprise features will continue to be available.

Trial Licenses

You can use a trial license only once; when the 60-day trial period expires or you have ended the trial, you cannot restart the trial.

When a trial ends, enterprise features immediately become unavailable. However, data or configurations associated with the disabled functions are not deleted, and become available again once you install a Cloudera Enterprise license.

Ending a Cloudera Enterprise Data Hub Edition Trial

If you are using the trial edition the License page indicates when your license will expire. However, you can end the trial at any time (prior to expiration) as follows:

1. On the License page, click **End Trial**.
2. Confirm that you want to end the trial.
3. Restart the Cloudera Management Service, HBase, HDFS, and Hive services to pick up configuration changes.

Upgrading from Cloudera Express to a Cloudera Enterprise Data Hub Edition Trial

To start a trial, on the License page, click **Try Cloudera Enterprise Data Hub Edition for 60 Days**.

1. Cloudera Manager displays a pop-up describing the features enabled with Cloudera Enterprise Data Hub Edition. Click **OK** to proceed. At this point, your installation is upgraded and the Customize Role Assignments page displays.
2. Under **Reports Manager** click **Select a host**. The pageable host selection dialog box displays.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
 - Rack name
3. Select a host and click **OK**.
 4. When you are satisfied with the assignments, click **Continue**.
 5. Configure database settings:
 - a. Choose the database type:
 - Keep the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Record the auto-generated passwords.

Cluster Setup

Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the [Installing and Configuring an External Database](#) section of the [Installation Guide](#).

Use Custom Databases
 Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

Service	Status
Hive Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="hive"/> Username: <input type="text" value="hive"/> Password: <input type="text" value="t56iwbdk4F"/>	✓ Skipped. Cloudera Manager will create this database in a later step.
Reports Manager Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="rman"/> Username: <input type="text" value="rman"/> Password: <input type="text" value="Y6S4iWvNo"/>	✓ Successful
Navigator Audit Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="nav"/> Username: <input type="text" value="nav"/> Password: <input type="text" value="QLR2B0qqO9"/>	✓ Successful
Navigator Metadata Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="navms"/> Username: <input type="text" value="navms"/> Password: <input type="text" value="lmo07jxOen"/>	✓ Successful
Oozie Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="oozie_oozie_se"/> Username: <input type="text" value="oozie_oozie_se"/> Password: <input type="text" value="NTF1KNdpPI"/>	✓ Skipped. Cloudera Manager will create this database in a later step.

[Test Connection](#)

- Select **Use Custom Databases** to specify external databases.
 1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database.
 - If you are adding the Oozie service, you can change your Oozie configuration to control when data is purged in order to improve performance, cut down on database disk usage, improve upgrade performance, or to keep the history for a longer period of time. See [Configuring Oozie Data Purge Settings Using Cloudera Manager](#) on page 209.
 - b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise, check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes screen displays.
6. Review the configuration changes to be applied. Confirm the settings entered for file system paths. The file paths required vary based on the services to be installed. If you chose to add the Sqoop service, indicate whether to use the default Derby database or the embedded PostgreSQL database. If the latter, type the database name, host, and user credentials that you specified when you created the database.



Warning: Do not place DataNode data directories on NAS devices. When resizing an NAS, block replicas can be deleted, which will result in reports of missing blocks.

Click **Continue**. The wizard starts the services.

7. At this point, your installation is upgraded. Click **Continue**.

- Restart Cloudera Management Services and audited services to pick up configuration changes. The audited services will write audit events to a log file, but the events are not transferred to the Cloudera Navigator Audit Server until you add and start the Cloudera Navigator Audit Server role as described in [Adding Cloudera Navigator Roles](#) on page 433. For information on Cloudera Navigator, see [Cloudera Navigator documentation](#).

Upgrading from a Cloudera Enterprise Data Hub Edition Trial to Cloudera Enterprise

- Purchase a Cloudera Enterprise license from Cloudera.
- On the License page, click **Upload License**.
- Click the document icon to the left of the **Select a License File** text field.
- Go to the location of your license file, click the file, and click **Open**.
- Click **Upload**.

Upgrading from Cloudera Express to Cloudera Enterprise

- Purchase a Cloudera Enterprise license from Cloudera.
- On the License page, click **Upload License**.
- Click the document icon to the left of the **Select a License File** text field.
- Go to the location of your license file, click the file, and click **Open**.
- Click **Upload**.
- Cloudera Manager displays a pop-up describing the features enabled with Cloudera Enterprise Data Hub Edition. Click **OK** to proceed. At this point, your installation is upgraded and the Customize Role Assignments page displays.
- Under **Reports Manager** click **Select a host**. The pageable host selection dialog displays.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

- When you are satisfied with the assignments, click **Continue**.
- Configure database settings:
 - Choose the database type:
 - Keep the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Record the auto-generated passwords.

Cluster Setup

Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the [Installing and Configuring an External Database](#) section of the [Installation Guide](#).

Use Custom Databases
 Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

Service	Status
Hive Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="hive"/> Username: <input type="text" value="hive"/> Password: <input type="text" value="t56iwbdk4F"/>	✓ Skipped. Cloudera Manager will create this database in a later step.
Reports Manager Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="rman"/> Username: <input type="text" value="rman"/> Password: <input type="text" value="Y6S4iWvNo"/>	✓ Successful
Navigator Audit Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="nav"/> Username: <input type="text" value="nav"/> Password: <input type="text" value="QLR2B0qqO9"/>	✓ Successful
Navigator Metadata Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="navms"/> Username: <input type="text" value="navms"/> Password: <input type="text" value="lmo07jxOen"/>	✓ Successful
Oozie Server Currently assigned to run on tcdn2-1.ent.cloudera.com . Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="oozie_oozie_se"/> Username: <input type="text" value="oozie_oozie_se"/> Password: <input type="text" value="NTF1KNdpPI"/>	✓ Skipped. Cloudera Manager will create this database in a later step.

[Test Connection](#)

- Select **Use Custom Databases** to specify external databases.
 1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database.
 - If you are adding the Oozie service, you can change your Oozie configuration to control when data is purged in order to improve performance, cut down on database disk usage, improve upgrade performance, or to keep the history for a longer period of time. See [Configuring Oozie Data Purge Settings Using Cloudera Manager](#) on page 209.
 - b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise, check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes screen displays.
- 10 Review the configuration changes to be applied. Confirm the settings entered for file system paths. The file paths required vary based on the services to be installed. If you chose to add the Sqoop service, indicate whether to use the default Derby database or the embedded PostgreSQL database. If the latter, type the database name, host, and user credentials that you specified when you created the database.



Warning: Do not place DataNode data directories on NAS devices. When resizing an NAS, block replicas can be deleted, which will result in reports of missing blocks.

Click **Continue**. The wizard starts the services.

- 11 At this point, your installation is upgraded. Click **Continue**.

- 12 Restart Cloudera Management Services and audited services to pick up configuration changes. The audited services will write audit events to a log file, but the events are not transferred to the Cloudera Navigator Audit Server until you add and start the Cloudera Navigator Audit Server role as described in [Adding Cloudera Navigator Roles](#) on page 433. For information on Cloudera Navigator, see [Cloudera Navigator documentation](#).

If you want to use the Cloudera Navigator Metadata Server, add its role following the instructions in [Adding Cloudera Navigator Roles](#) on page 433.

Renewing a License

1. Download the license file and save it locally.
2. In Cloudera Manager, go to the **Home** page.
3. Select **Administration > License**.
4. Click **Upload License**.
5. Browse to the license file you downloaded.
6. Click **Upload**.

You do not need to restart Cloudera Manager for the new license to take effect.

Sending Usage and Diagnostic Data to Cloudera

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

Cloudera Manager collects anonymous usage information and takes regularly-scheduled snapshots of the state of your cluster and automatically sends them anonymously to Cloudera. This helps Cloudera improve and optimize Cloudera Manager.

If you have a Cloudera Enterprise license, you can also trigger the collection of diagnostic data and send it to Cloudera Support to aid in resolving a problem you may be having.

Configuring a Proxy Server

To configure a proxy server through which usage and diagnostic data is uploaded, follow the instructions in [Configuring Network Settings](#) on page 418.

Managing Anonymous Usage Data Collection

Cloudera Manager sends anonymous usage information using Google Analytics to Cloudera. The information helps Cloudera improve Cloudera Manager. By default anonymous usage data collection is *enabled*.

1. Select **Administration > Settings**.
2. Under the **Other** category, set the **Allow Usage Data Collection** property.
3. Click **Save Changes** to commit the changes.

Managing Hue Analytics Data Collection

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

Hue tracks anonymized pages and application versions to collect information used to compare each application's usage levels. The data collected does not include hostnames or IDs; For example, the data has the format /2.3.0/pig, /2.5.0/beeswax/execute. You can restrict data collection as follows:

1. Go to the Hue service.
2. Click the **Configuration** tab.
3. Select **Scope > Hue**.
4. Locate the **Enable Usage Data Collection** property or search for it by typing its name in the Search box.
5. Deselect the **Enable Usage Data Collection** checkbox.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

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

- Restart the Hue service.

Diagnostic Data Collection

To help with solving problems when using Cloudera Manager on your cluster, Cloudera Manager collects diagnostic data on a regular schedule, and automatically sends it to Cloudera. By default Cloudera Manager is configured to collect data weekly and to send it *automatically*. You can schedule the frequency of data collection on a daily, weekly, or monthly schedule, or disable the scheduled collection of data entirely. You can also send a collected data set [manually](#).



Note:

- Automatically sending diagnostic data requires the Cloudera Manager Server host to have Internet access, and be configured for sending data automatically. If your Cloudera Manager server does not have Internet access, and you have a Cloudera Enterprise license, you can manually send the diagnostic data as described in [Manually Triggering Collection and Transfer of Diagnostic Data to Cloudera](#) on page 428.
- Automatically sending diagnostic data may fail sometimes and return an error message of "Could not send data to Cloudera." To work around this issue, you can manually send the data to Cloudera Support.

What Data Does Cloudera Manager Collect?

Cloudera Manager collects and returns a significant amount of information about the health and performance of the cluster. It includes:

- Up to 1000 Cloudera Manager audit events: Configuration changes, add/remove of users, roles, services, etc.
- One day's worth of Cloudera Manager events: This includes critical errors Cloudera Manager watches for and more
- Data about the cluster structure which includes a list of all hosts, roles, and services along with the configurations that are set through Cloudera Manager. Where passwords are set in Cloudera Manager, the passwords are not returned.
- Cloudera Manager license and version number.
- Current health information for hosts, service, and roles. Includes results of health tests run by Cloudera Manager.
- Heartbeat information from each host, service, and role. These include status and some information about memory, disk, and processor usage.
- The results of running Host Inspector.
- One day's worth of Cloudera Manager metrics.



Note: If you are using Cloudera Express, host metrics are not included.

- A download of the debug pages for Cloudera Manager roles.
- For each host in the cluster, the result of running a number of system-level commands on that host.
- Logs from each role on the cluster, as well as the Cloudera Manager server and agent logs.
- Which parcels are activated for which clusters.
- Whether there's an active trial, and if so, metadata about the trial.
- Metadata about the Cloudera Manager server, such as its JMX metrics, stack traces, and the database/host it's running with.
- HDFS/Hive replication schedules (including command history) for the deployment.
- Impala query logs.

Configuring the Frequency of Diagnostic Data Collection

By default, Cloudera Manager collects diagnostic data on a weekly basis. You can change the frequency to daily, weekly, monthly, or never. If you are a Cloudera Enterprise customer and you set the schedule to **never** you can still collect and send data to Cloudera on demand. If you are a Cloudera Express customer and you set the schedule to **never**, data is not collected or sent to Cloudera.

1. Select **Administration > Settings**.
2. Under the **Support** category, click **Scheduled Diagnostic Data Collection Frequency** and select the frequency.
3. To set the day and time of day that the collection will be performed, click **Scheduled Diagnostic Data Collection Time** and specify the date and time in the pop-up control.
4. Click **Save Changes** to commit the changes.

You can see the current setting of the data collection frequency by viewing **Support > Scheduled Diagnostics**: in the main navigation bar.

Specifying the Diagnostic Data Directory

You can configure the directory where collected data is stored.

1. Select **Administration > Settings**.
2. Under the **Support** category, set the **Diagnostic Data Bundle Directory** to a directory on the host running Cloudera Manager Server. The directory must exist and be enabled for writing by the user `cloudera-scm`. If this field is left blank, the data is stored in `/tmp`.
3. Click **Save Changes** to commit the changes.

Collecting and Sending Diagnostic Data to Cloudera



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Disabling the Automatic Sending of Diagnostic Data from a Manually Triggered Collection

If you do not want data automatically sent to Cloudera after manually triggering data collection, you can disable this feature. The data you collect will be saved and can be downloaded for sending to Cloudera Support at a later time.

1. Select **Administration > Settings**.
2. Under the **Support** category, uncheck the box for **Send Diagnostic Data to Cloudera Automatically**.
3. Click **Save Changes** to commit the changes.



Note: The Send Diagnostic Data form that displays when you collect data in one of the following procedures indicates whether the data will be sent automatically.

Manually Triggering Collection and Transfer of Diagnostic Data to Cloudera

1. Optionally change the System Identifier property:
 - a. Select **Administration > Settings**.
 - b. Under the **Other** category, set the System Identifier property and click **Save Changes**.
2. Under the **Support** menu at the top right of the navigation bar, choose **Send Diagnostic Data**. The Send Diagnostic Data form displays.
3. Fill in or change the information here as appropriate:
 - Optionally, you can improve performance by reducing the size of the data bundle that is sent. Click **Restrict log and metrics collection** to expand this section of the form. The three filters, **Host**, **Service**, and **Role Type**,

allow you to restrict the data that will be sent. Cloudera Manager will only collect logs and metrics for roles that match all three filters.

- Cloudera Manager populates the **End Time** based on the setting of the Time Range selector. You should change this to be a few minutes after you observed the problem or condition that you are trying to capture. The time range is based on the timezone of the host where Cloudera Manager Server is running.
- If you have a support ticket open with Cloudera Support, include the support ticket number in the field provided.

4. Depending on whether you have disabled automatic sending of data, do one of the following:

- Click **Collect and Send Diagnostic Data**. A Running Commands window shows you the progress of the data collection steps. When these steps are complete, the collected data is sent to Cloudera.
- Click **Collect Diagnostic Data**. A Command Details window shows you the progress of the data collection steps.
 1. In the Command Details window, click **Download Result Data** to download and save a zip file of the information.
 2. Send the data to Cloudera Support by doing one of the following:
 - Send the bundle using a Python script:
 1. Download the [phone_home](#) script.
 2. Copy the script and the downloaded data file to a host that has Internet access.
 3. Run the following command on that host:

```
python phone_home.py --file downloaded_data_file
```

- Attach the bundle to the SFDC case. Do not rename the bundle as this can cause a delay in processing the bundle.
- Contact [Cloudera Support](#) and arrange to send the data file.

Exporting and Importing Cloudera Manager Configuration

You can use the Cloudera Manager API to programmatically export and import a definition of all the entities in your Cloudera Manager-managed deployment—clusters, service, roles, hosts, users and so on. See the [Cloudera Manager API](#) documentation on how to manage deployments using the [/cm/deployment](#) resource.

Other Cloudera Manager Tasks and Settings

From the **Administration** tab you can select options for configuring settings that affect how Cloudera Manager interacts with your clusters.

Settings

The **Settings** page provides a number of categories as follows:

- **Performance** - Set the Cloudera Manager Agent heartbeat interval. See [Configuring Agent Heartbeat and Health Status Options](#) on page 412.
- **Advanced** - Enable API debugging and other advanced options.
- **Monitoring** - Set Agent health status parameters. For configuration instructions, see [Configuring Cloudera Manager Agents](#) on page 412.
- **Security** - Set TLS encryption settings to enable TLS encryption between the Cloudera Manager Server, Agents, and clients. For configuration instructions, see [Configuring TLS Security for Cloudera Manager](#). You can also:
 - Set the realm for Kerberos security and point to a custom keytab retrieval script. For configuration instructions, see [Cloudera Security](#).
 - Specify session timeout and a "Remember Me" option.

- **Ports and Addresses** - Set ports for the Cloudera Manager Admin Console and Server. For configuration instructions, see [Configuring Cloudera Manager Server Ports](#) on page 410.
- **Other**
 - Enable Cloudera usage data collection For configuration instructions, see [Managing Anonymous Usage Data Collection](#) on page 426.
 - Set a custom header color and banner text for the Admin console.
 - Set an "Information Assurance Policy" statement – this statement will be presented to every user before they are allowed to access the login dialog box. The user must click "I Agree" in order to proceed to the login dialog box.
 - Disable/enable the auto-search for the Events panel at the bottom of a page.
- **Support**
 - Configure diagnostic data collection properties. See [Diagnostic Data Collection](#) on page 427.
 - Configure how to access Cloudera Manager [help](#) files.
- **External Authentication** - Specify the configuration to use LDAP, Active Directory, or an external program for authentication. See [Configuring External Authentication for Cloudera Manager](#) for instructions.
- **Parcels** - Configure settings for parcels, including the location of remote repositories that should be made available for download, and other settings such as the frequency with which Cloudera Manager will check for new parcels, limits on the number of downloads or concurrent distribution uploads. See [Parcels](#) for more information.
- **Network** - Configure proxy server settings. See [Configuring Network Settings](#) on page 418.
- **Custom Service Descriptors** - Configure custom service descriptor properties for [Add-on Services](#) on page 37.

Alerts

See [Managing Alerts](#) on page 419.

Users

See [Cloudera Manager User Accounts](#).

Kerberos

See [Enabling Kerberos Authentication Using the Wizard](#).

License

See [Managing Licenses](#) on page 421.

User Interface Language

You can change the language of the Cloudera Manager Admin Console User Interface through the language preference in your browser. Information on how to do this for the browsers supported by Cloudera Manager is shown under the Administration page. You can also change the language for the information provided with activity and health events, and for alert email messages by selecting **Language**, selecting the language you want from the drop-down list on this page, then clicking **Save Changes**.

Peers

See [Designating a Replication Source](#) on page 379.

Cloudera Management Service

The Cloudera Management Service implements various management features as a set of roles:

- **Activity Monitor** - collects information about activities run by the MapReduce service. This role is not added by default.
- **Host Monitor** - collects health and metric information about hosts

- Service Monitor - collects health and metric information about services and activity information from the YARN and Impala services
- Event Server - aggregates relevant Hadoop events and makes them available for alerting and searching
- Alert Publisher - generates and delivers alerts for certain types of events
- Reports Manager - generates reports that provide an historical view into disk utilization by user, user group, and directory, processing activities by user and YARN pool, and HBase tables and namespaces. This role is not added in Cloudera Express.

Cloudera Manager manages each role separately, instead of as part of the Cloudera Manager Server, for scalability (for example, on large deployments it's useful to put the monitor roles on their own hosts) and isolation.

In addition, for certain editions of the Cloudera Enterprise license, the Cloudera Management Service provides the [Navigator Audit Server](#) and [Navigator Metadata Server](#) roles for [Cloudera Navigator](#).

Displaying the Cloudera Management Service Status

1. Do one of the following:
 - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

Starting the Cloudera Management Service

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

1. Do one of the following:
 - 1. Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
 - 2. Select **Actions** > **Start**.
 - 1. On the Home page, click

 to the right of **Cloudera Management Service** and select **Start**.
2. Click **Start** to confirm. The **Command Details** window shows the progress of starting the roles.
3. When **Command completed with n/n successful subcommands** appears, the task is complete. Click **Close**.

Stopping the Cloudera Management Service

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

1. Do one of the following:
 - 1. Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
 - 2. Select **Actions** > **Stop**.
 - 1. On the Home page, click

 to the right of **Cloudera Management Service** and select **Stop**.
2. Click **Stop** to confirm. The **Command Details** window shows the progress of stopping the roles.
3. When **Command completed with n/n successful subcommands** appears, the task is complete. Click **Close**.

Restarting the Cloudera Management Service

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

1. Do one of the following:

- **1. Select Clusters > Cloudera Management Service > Cloudera Management Service.**
- **2. Select Actions > Restart.**

- On the Home page, click



to the right of **Cloudera Management Service** and select **Restart**.

2. Click **Restart** to confirm. The **Command Details** window shows the progress of stopping and then starting the roles.
3. When **Command completed with n/n successful subcommands** appears, the task is complete. Click **Close**.

Starting and Stopping Cloudera Management Service Roles

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

1. Do one of the following:

- Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
- On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the **Instances** tab.

3. Check the checkbox next to a role.

4. Do one of the following depending on your user role:

- **Minimum Required Role:** [Full Administrator](#)

Choose and action:

- Select **Actions for Selected > Start** and click **Start** to confirm.
- Select **Actions for Selected > Stop** and click **Stop** to confirm.

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

1. Click a Cloudera Navigator Audit Server or Cloudera Navigator Metadata Server link.
2. Choose an action:

- Select **Actions > Start this XXX** and click **Start this XXX** to confirm, where XXX is the role name.
- Select **Actions > Stop this XXX** and click **Stop this XXX** to confirm, where XXX is the role name.

Configuring Management Service Database Limits

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

Each Cloudera Management Service role maintains a database for retaining the data it monitors. These databases (as well as the log files maintained by these services) can grow quite large. For example, the Activity Monitor maintains data at the service level, the activity level (MapReduce jobs and aggregate activities), and at the task attempt level. Limits on these data sets are configured when you create the management services, but you can modify these parameters through the Configuration settings in the Cloudera Manager Admin Console. For example, the Event Server lets you set a total number of events to store, and Activity Monitor gives you "purge" settings (also in hours) for the data it stores.

There are also settings for the logs that these various services create. You can throttle how big the logs are allowed to get and how many previous logs to retain.

1. Do one of the following:

- Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
- On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the **Configuration** tab.
3. Select **Scope** and then one of the following.
 - **Activity Monitor** - the **Purge** or **Expiration** period properties are found in the top-level settings for the role.
 - **Host Monitor** - see [Data Storage for Monitoring Data](#).
 - **Service Monitor**
4. Select **Category** > **Log Files** to view log file size properties.
5. Edit the appropriate properties.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
6. Click **Save Changes**.

Adding Cloudera Navigator Roles

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

1. Do one of the following:
 - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button. The Customize Role Assignments page displays.
4. Assign the Navigator role to a host.
 - a. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

5. When you are satisfied with the assignments, click **Continue**.
6. Configure database settings:
 - a. Choose the database type:
 - Keep the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Record the auto-generated passwords.

Cluster Setup

Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the [Installing and Configuring an External Database](#) section of the [Installation Guide](#).

Use Custom Databases
 Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

Service	Status
Hive Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="hive"/> Username: <input type="text" value="hive"/> Password: <input type="text" value="t56iwbdk4F"/>	✓ Skipped. Cloudera Manager will create this database in a later step.
Reports Manager Currently assigned to run on tcdn2-1.ent.cloudera.com. Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="rman"/> Username: <input type="text" value="rman"/> Password: <input type="text" value="Y6S4iWvNo"/>	✓ Successful
Navigator Audit Server Currently assigned to run on tcdn2-1.ent.cloudera.com. Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="nav"/> Username: <input type="text" value="nav"/> Password: <input type="text" value="QLR2B0qqO9"/>	✓ Successful
Navigator Metadata Server Currently assigned to run on tcdn2-1.ent.cloudera.com. Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="navms"/> Username: <input type="text" value="navms"/> Password: <input type="text" value="lmo07jxOen"/>	✓ Successful
Oozie Server Currently assigned to run on tcdn2-1.ent.cloudera.com. Database Host Name: <input type="text" value="tcdn2-1.ent.cloudera.com:7432"/> Database Type: PostgreSQL Database Name: <input type="text" value="oozie_oozie_se"/> Username: <input type="text" value="oozie_oozie_se"/> Password: <input type="text" value="NTF1KNdpPI"/>	✓ Skipped. Cloudera Manager will create this database in a later step.

[Test Connection](#)

- Select **Use Custom Databases** to specify external databases.
 1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database.
- If you are adding the Oozie service, you can change your Oozie configuration to control when data is purged in order to improve performance, cut down on database disk usage, improve upgrade performance, or to keep the history for a longer period of time. See [Configuring Oozie Data Purge Settings Using Cloudera Manager](#) on page 209.
- b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise, check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes screen displays.

7. Click **Finish**.

Deleting Cloudera Navigator Roles

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

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Instances** tab.

3. Check the checkboxes next to the **Navigator Audit Server** and **Navigator Metadata Server** roles.
4. Do one of the following depending on your role:
 - **Minimum Required Role: [Full Administrator](#)**
 1. Check the checkboxes next to the **Navigator Audit Server** and **Navigator Metadata Server** roles.
 2. Select **Actions for Selected > Stop** and click **Stop** to confirm.
 - **Minimum Required Role: [Navigator Administrator](#)** (also provided by **Full Administrator**)
 1. Click the **Navigator Audit Server** role link.
 2. Select **Actions > Stop this Navigator Audit Server** and click **Stop this Navigator Audit Server** to confirm.
 3. Click the **Navigator Metadata Server** role link.
 4. Select **Actions > Stop this Navigator Metadata Server** and click **Stop this Navigator Metadata Server** to confirm.
5. Check the checkboxes next to the **Navigator Audit Server** and **Navigator Metadata Server** roles.
6. Select **Actions for Selected > Delete**. Click **Delete** to confirm the deletion.

Cloudera Navigator Data Management Component Administration

The Cloudera Navigator data management component is implemented as two roles within the [Cloudera Management Service](#) on page 430.

For information on managing the Cloudera Navigator data management roles, see the following topics.

Related Information

- [Cloudera Navigator 2 Overview](#)
- [Installing the Cloudera Navigator Data Management Component](#)
- [Upgrading the Cloudera Navigator Data Management Component](#)
- [Cloudera Data Management](#)
- [Configuring Authentication in the Cloudera Navigator Data Management Component](#)
- [Configuring SSL for the Cloudera Navigator Data Management Component](#)
- [Cloudera Navigator Data Management Component User Roles](#)

Cloudera Navigator Audit Server

The Navigator Audit Server performs the following functions:

- Tracking and coalescing events
- Storing events to the audit database



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Adding the Navigator Audit Server Role

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

1. Configure [the database](#) where audit events are stored.
2. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
3. Click the **Instances** tab.
4. Click the **Add Role Instances** button. The Customize Role Assignments page displays.
5. Assign the Navigator role to a host.
 - a. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

- Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

6. When you are satisfied with the assignments, click **Continue**.

7. Configure database settings:

a. Choose the database type:

- Keep the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Record the auto-generated passwords.

Cluster Setup

Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the [Installing and Configuring an External Database](#) section of the [Installation Guide](#).

Use Custom Databases
 Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

Hive ✔ Skipped. Cloudera Manager will create this database in a later step.					
Database Host Name:	Database Type:	Database Name :	Username:	Password:	
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="hive"/>	<input type="text" value="hive"/>	<input type="text" value="t56iwbdk4F"/>	
Reports Manager ✔ Successful					
Currently assigned to run on tcdn2-1.ent.cloudera.com.					
Database Host Name:	Database Type:	Database Name :	Username:	Password:	
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="rman"/>	<input type="text" value="rman"/>	<input type="text" value="Y6S4iWvfn0"/>	
Navigator Audit Server ✔ Successful					
Currently assigned to run on tcdn2-1.ent.cloudera.com.					
Database Host Name:	Database Type:	Database Name :	Username:	Password:	
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="nav"/>	<input type="text" value="nav"/>	<input type="text" value="QLR2B0qqO9"/>	
Navigator Metadata Server ✔ Successful					
Currently assigned to run on tcdn2-1.ent.cloudera.com.					
Database Host Name:	Database Type:	Database Name :	Username:	Password:	
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="navms"/>	<input type="text" value="navms"/>	<input type="text" value="lmo07jx0en"/>	
Oozie Server ✔ Skipped. Cloudera Manager will create this database in a later step.					
Currently assigned to run on tcdn2-1.ent.cloudera.com.					
Database Host Name:	Database Type:	Database Name :	Username:	Password:	
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="oozie_oozie_se"/>	<input type="text" value="oozie_oozie_se"/>	<input type="text" value="NTF1KNdpPI"/>	

- Select **Use Custom Databases** to specify external databases.
 1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database.
- If you are adding the Oozie service, you can change your Oozie configuration to control when data is purged in order to improve performance, cut down on database disk usage, improve upgrade performance,

or to keep the history for a longer period of time. See [Configuring Oozie Data Purge Settings Using Cloudera Manager](#) on page 209.

- b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise, check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes screen displays.

8. Click **Finish**.

Starting, Stopping, and Restarting the Navigator Audit Server

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Instances** tab.
3. Do one of the following depending on your role:
 - **Minimum Required Role: [Full Administrator](#)**
 1. Select the checkbox next to **Navigator Audit Server**.
 2. Select **Actions for Selected > Action**. Click *Action* to confirm the action, where *Action* is Start, Stop, or Restart.
 - **Minimum Required Role: [Navigator Administrator](#)** (also provided by **Full Administrator**)
 1. Click the **Navigator Audit Server** role link.
 2. Select **Actions > Action this Navigator Audit Server**. Click *Action this Navigator Audit Server*, where *Action* is Start, Stop, or Restart, to confirm the action.

Configuring the Navigator Audit Server Log Directory

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

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Audit Server**.
4. Select **Category > Logs**.
5. Set the **Navigator Audit Server Log Directory** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
6. Click **Save Changes**.
7. Click the **Instances** tab.
8. Restart the role.

Configuring the Navigator Audit Server Data Expiration Period

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

You can configure the number of hours of audit events to keep in the Navigator Audit Server database as follows:

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Audit Server**.
4. Select **Category > All**.
5. Set the **Navigator Audit Server Data Expiration Period** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.
6. Restart the role.

Cloudera Navigator Metadata Server

The Navigator Metadata Server performs the following functions:

- Obtains connection information about CDH services from the Cloudera Manager Server
- Extracts metadata for the entities managed by those services at periodic intervals
- Manages and applies [metadata extraction policies](#) during metadata extraction
- Indexes and stores entity metadata
- Manages authorization data for Navigator users
- Manages [audit report](#) metadata
- Implements the Navigator UI and API



Important: This feature is available only with a Cloudera Enterprise license; it is not available in Cloudera Express. For information on Cloudera Enterprise licenses, see [Managing Licenses](#) on page 421.

Adding the Navigator Metadata Server

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

1. Configure [the database](#) where policies, roles, and audit report metadata is stored.
2. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
3. Click the **Instances** tab.
4. Click the **Add Role Instances** button. The Customize Role Assignments page displays.
5. Assign the Navigator role to a host.
 - a. Customize the assignment of role instances to hosts. The wizard evaluates the hardware configurations of the hosts to determine the best hosts for each role. The wizard assigns all worker roles to the same set of hosts to which the HDFS DataNode role is assigned. You can reassign role instances if necessary.

Click a field below a role to display a dialog containing a list of hosts. If you click a field containing multiple hosts, you can also select **All Hosts** to assign the role to all hosts, or **Custom** to display the pageable hosts dialog.

The following shortcuts for specifying hostname patterns are supported:

 - Range of hostnames (without the domain portion)

Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

- IP addresses
- Rack name

Click the **View By Host** button for an overview of the role assignment by hostname ranges.

6. When you are satisfied with the assignments, click **Continue**.

7. Configure database settings:

a. Choose the database type:

- Keep the default setting of **Use Embedded Database** to have Cloudera Manager create and configure required databases. Record the auto-generated passwords.

Cluster Setup

Database Setup

Configure and test database connections. If using custom databases, create the databases first according to the [Installing and Configuring an External Database](#) section of the [Installation Guide](#).

Use Custom Databases
 Use Embedded Database

When using the embedded database, passwords are automatically generated. Please copy them down.

Hive ✔ Skipped. Cloudera Manager will create this database in a later step.

Currently assigned to run on `tcdn2-1.ent.cloudera.com`.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="hive"/>	<input type="text" value="hive"/>	<input type="text" value="t56iwbdk4F"/>

Reports Manager ✔ Successful

Currently assigned to run on `tcdn2-1.ent.cloudera.com`.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="rman"/>	<input type="text" value="rman"/>	<input type="text" value="Y6S4iWvfn0"/>

Navigator Audit Server ✔ Successful

Currently assigned to run on `tcdn2-1.ent.cloudera.com`.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="nav"/>	<input type="text" value="nav"/>	<input type="text" value="QLR2B0qqO9"/>

Navigator Metadata Server ✔ Successful

Currently assigned to run on `tcdn2-1.ent.cloudera.com`.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="navms"/>	<input type="text" value="navms"/>	<input type="text" value="lmo07jxOen"/>

Oozie Server ✔ Skipped. Cloudera Manager will create this database in a later step.

Currently assigned to run on `tcdn2-1.ent.cloudera.com`.

Database Host Name:	Database Type:	Database Name :	Username:	Password:
<input type="text" value="tcdn2-1.ent.cloudera.com:7432"/>	<input type="text" value="PostgreSQL"/>	<input type="text" value="oozie_oozie_se"/>	<input type="text" value="oozie_oozie_se"/>	<input type="text" value="NTF1KNdpPI"/>

- Select **Use Custom Databases** to specify external databases.
 1. Enter the database host, database type, database name, username, and password for the database that you created when you set up the database.
- If you are adding the Oozie service, you can change your Oozie configuration to control when data is purged in order to improve performance, cut down on database disk usage, improve upgrade performance,

or to keep the history for a longer period of time. See [Configuring Oozie Data Purge Settings Using Cloudera Manager](#) on page 209.

- b. Click **Test Connection** to confirm that Cloudera Manager can communicate with the database using the information you have supplied. If the test succeeds in all cases, click **Continue**; otherwise, check and correct the information you have provided for the database and then try the test again. (For some servers, if you are using the embedded database, you will see a message saying the database will be created at a later step in the installation process.) The Review Changes screen displays.

8. Click Finish.

Starting, Stopping, and Restarting the Navigator Metadata Server

1. Do one of the following:

- Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
- On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the Instances tab.

3. Do one of the following depending on your role:

- **Minimum Required Role: [Full Administrator](#)**
 1. Select the checkbox next to the **Navigator Metadata Server** role.
 2. Select **Actions for Selected > Action**. Click *Action* to confirm the action, where *Action* is Start, Stop, or Restart.
- **Minimum Required Role: [Navigator Administrator](#)** (also provided by **Full Administrator**)
 1. Click the **Navigator Metadata Server** role link.
 2. Select **Actions > Action this Navigator Metadata Server**. Click *Action this Navigator Metadata Server*, where *Action* is Start, Stop, or Restart, to confirm the action.

Navigator Metadata Server Sizing and Performance Recommendations

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

Two activities determine Navigator Metadata Server resource requirements:

- Extracting metadata from the cluster and creating relationships
- Querying

The Navigator Metadata Server uses Solr to store, index, and query metadata. Indexing happens during extraction. Querying is fast and efficient because the data is indexed.

Memory and CPU requirements are based on amount of data that is stored and indexed. With 6 GB of RAM and 8-10 cores Solr can process 6 million entities in 25-30 minutes or 80 million entities in 8 to 9 hours. Any less RAM than 6GB and will result in excessive garbage collection and possibly out-of-memory exceptions. For large clusters, Cloudera advises at least 8 GB of RAM and 8 cores. The Solr instance runs in process with Navigator, so the Java heap for the Navigator Metadata Server should be set according to the size of cluster.

By default, during the Cloudera Manager Installation wizard the Navigator Audit Server and Navigator Metadata Server are assigned to the same host as the Cloudera Management Service monitoring roles. This configuration works for a small cluster, but should be updated before the cluster grows. You can either change the configuration at installation time or move the Navigator Metadata Server if necessary.

Moving the Navigator Metadata Server

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

1. Stop the Navigator Metadata Server, delete it from existing host, and add it to a new host.

Cloudera Navigator Data Management Component Administration

2. If the Solr data path is not on NFS/SAN, move the data to the same path on the new host.
3. Start the Navigator Metadata Server.

Configuring the Navigator Metadata Server Storage Directory

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

The section describes how to configure where the Navigator Metadata Server stores extracted and indexed data. The default is `/var/lib/cloudera-scm-navigator`.

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Metadata Server**.
4. Specify the directory in the **Navigator Metadata Server Storage Dir** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes**.
6. Click the **Instances** tab.
7. Restart the role.

Configuring the Navigator Metadata Server Port

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

Describes how to configure the port on which the Navigator UI is accessed. The default is 7187.

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. In the Search box, type `ports`.
4. Specify the port in the **Navigator Metadata Server Port** property.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes**.
6. Click the **Instances** tab.
7. Restart the role.

Enabling Hive Metadata Extraction in a Secure Cluster

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

The Navigator Metadata Server uses the hue user to connect to the Hive Metastore. The hue user is able to connect to the Hive Metastore by default. However, if the Hive service **Hive Metastore Access Control and Proxy User Groups Override** property and/or the HDFS service **Hive Proxy User Groups** property have been changed from their default values to settings that prevent the hue user from connecting to the Hive Metastore, Navigator Metadata Server will be unable to extract metadata from Hive. If this is the case, modify the Hive service **Hive Metastore Access Control and Proxy User Groups Override** property and/or the HDFS service **Hive Proxy User Groups** property so that the hue user can connect as follows:

1. Go to the Hive or HDFS service.
2. Click the **Configuration** tab.
3. In the Search box, type `proxy`.
4. In the Hive service **Hive Metastore Access Control and Proxy User Groups Override** or the HDFS service **Hive Proxy User Groups** property, click **+** to add a new row.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Type `hue`.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

Enabling Spark Metadata Extraction

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

Spark is an unsupported service and by default Spark metadata extraction is disabled. To enable Spark metadata extraction:

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Metadata Server**.
4. In **Navigator Metadata Server Advanced Configuration Snippet (Safety Valve) for cloudera-navigator.properties**, set the property

```
nav.spark.extraction.enable=true
```

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Configuring a JMS Server for Policy Messages

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

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Category > Policies**.
4. Set the following properties:

Property	Description
JMS URL	The URL of the JMS server to which notifications of changes to entities affected by policies are sent. Default: <code>tcp://localhost:61616</code> .

Property	Description
JMS User	The JMS user to which notifications of changes to entities affected by policies are sent. Default: admin.
JMS Password	The password of the JMS user to which notifications of changes to entities affected by policies are sent. Default: admin.
JMS Queue	The JMS queue to which notifications of changes to entities affected by policies are sent. Default: Navigator.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Enabling and Disabling Metadata Policy Expression Input

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

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Category > Policies**.
4. Select or deselect the **Enable Expression Input** checkbox.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Configuring the Session Timeout

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

The default session timeout is 30 minutes. To change the timeout period:

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Metadata Server**.
4. In **Navigator Metadata Server Advanced Configuration Snippet (Safety Valve) for cloudera-navigator.properties**, set the property

```
nav.max_inactive_interval=period (s)
```

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Managing Anonymous Usage Data Collection

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

Cloudera Navigator sends anonymous usage information using Google Analytics to Cloudera. The information helps Cloudera improve Cloudera Navigator. By default anonymous usage data collection is *enabled*. To enable and disable data collection:

1. Do one of the following:
 - Select **Clusters > Cloudera Management Service > Cloudera Management Service**.
 - On the Status tab of the Home page, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope > Navigator Metadata Server**.
4. Select or deselect the **Allow Usage Data Collection** checkbox.

If more than one role group applies to this configuration, edit the value for the appropriate role group. See [Modifying Configuration Properties](#) on page 10.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

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.
```