Cloudera Administration

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

Release Information

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

Table of Contents

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 that 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.

Server and Client Configuration

Administrators are sometimes surprised that modifying `/etc/hadoop/conf` and then restarting HDFS has no effect. That is because service instances started by Cloudera Manager do not read configurations from the default locations. To use HDFS as an example, when not managed by Cloudera Manager, there would usually be one HDFS configuration per host, located at `/etc/hadoop/conf/hdfs-site.xml`. Server-side daemons and clients running on the same host would all use that same configuration.

Cloudera Manager distinguishes between server and client configuration. In the case of HDFS, the file `/etc/hadoop/conf/hdfs-site.xml` contains only configuration relevant to an HDFS client. That is, by default, if you run a program that needs to communicate with Hadoop, it will get the addresses of the NameNode and JobTracker, and other important configurations, from that directory. A similar approach is taken for `/etc/hbase/conf` and `/etc/hive/conf`.

In contrast, the HDFS role instances (for example, NameNode and DataNode) obtain their configurations from a private per-process directory, under `/var/run/cloudera-scm-agent/process/`*unique-process-name*. Giving each process its own private execution and configuration environment allows Cloudera Manager to control each process independently. For example, here are the contents of an example `879-hdfs-NAMENODE` process directory:

```
$ tree -a /var/run/cloudera-scm-Agent/process/879-hdfs-NAMENODE/
  /var/run/cloudera-scm-Agent/process/879-hdfs-NAMENODE/
      cloudera_manager_Agent_fencer.py
      cloudera_manager_Agent_fencer_secret_key.txt
      cloudera-monitor.properties
      core-site.xml
      dfs_hosts_allow.txt
      dfs_hosts_exclude.txt
      event-filter-rules.json
      hadoop-metrics2.properties
      hdfs.keytab
      hdfs-site.xml
      log4j.properties
      logs
          stderr.log
          stdout.log
      topology.map
      topology.py
```

Distinguishing between server and client configuration provides several advantages:

- Sensitive information in the server-side configuration, such as the password for the Hive Metastore RDBMS, is not exposed to the clients.
- A service that depends on another service may deploy with customized configuration. For example, to get good HDFS read performance, Impala needs a specialized version of the HDFS client configuration, which may be harmful to a generic client. This is achieved by separating the HDFS configuration for the Impala daemons (stored in the per-process directory mentioned above) from that of the generic client (`/etc/hadoop/conf`).
- Client configuration files are much smaller and more readable. This also avoids confusing non-administrator Hadoop users with irrelevant server-side properties.

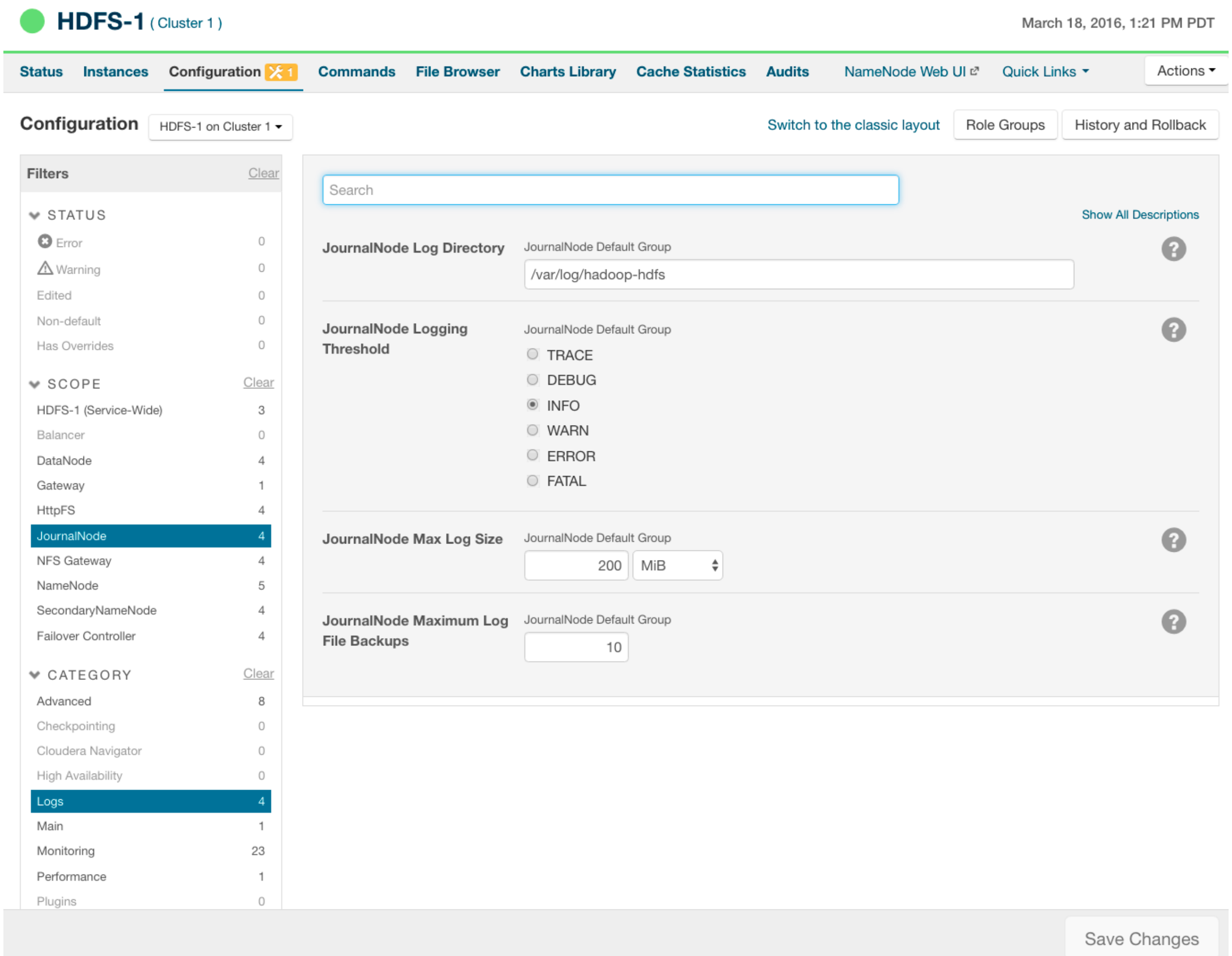
Cloudera Manager Configuration Layout

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.

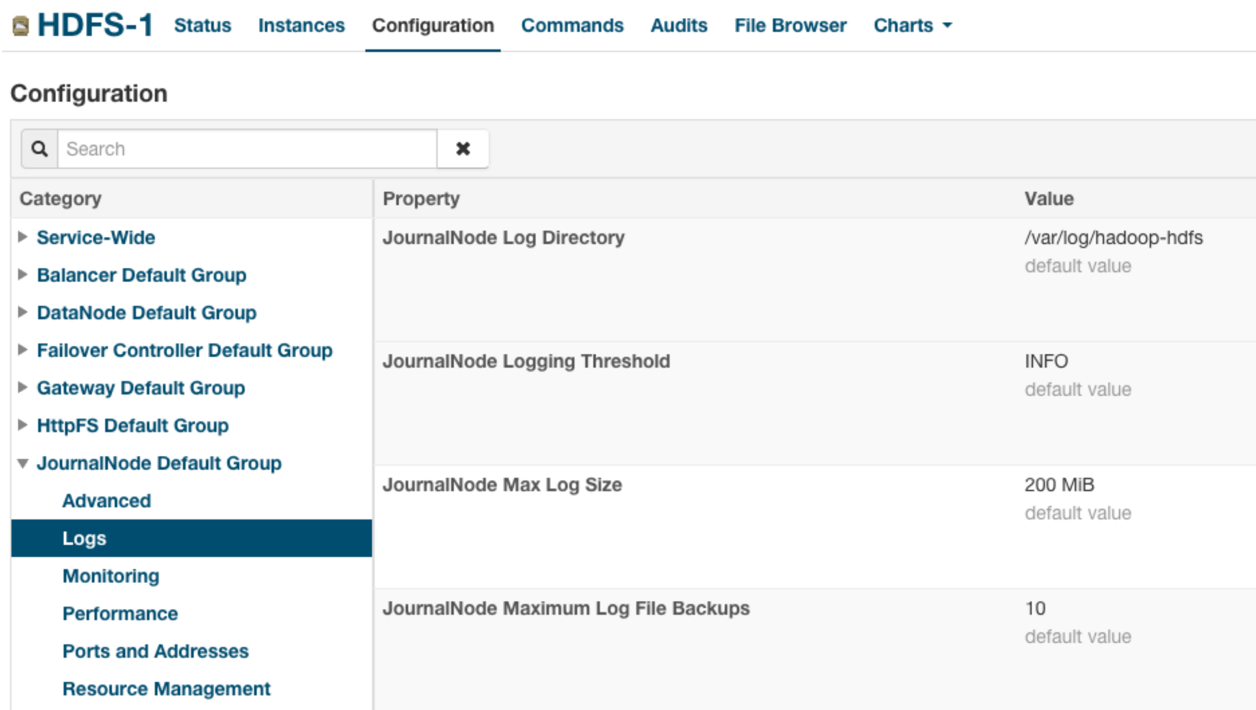
New layout pages contain controls that allow you to filter configuration properties based on configuration status, category, and group. For example, to display the JournalNode maximum log size property (JournalNode Max Log Size), click the **CATEGORY** > **JournalNode** and **GROUP** > **Logs** filters:



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**.



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 Using Cloudera Manager

> **Note:**
>
> This topic discusses how to configure properties using the Cloudera Manager "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 18.
>
> 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 57.

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.

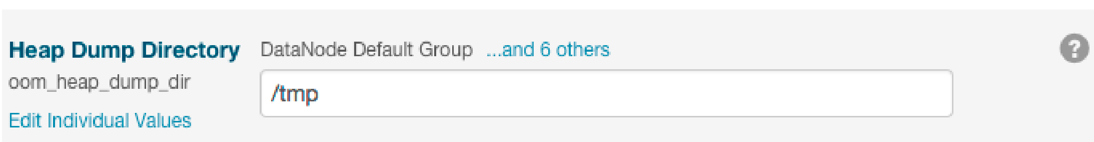
4. Edit the property value.

    - 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.

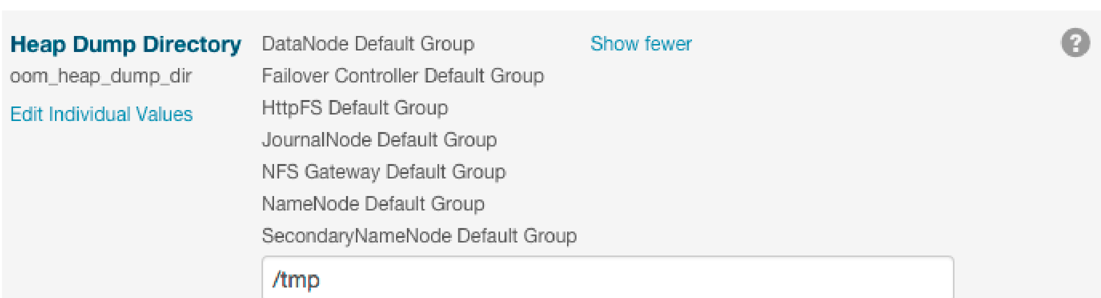
    Many configuration properties have different values that are configured by multiple role groups. (See [Role Groups](#) on page 57).

    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**.

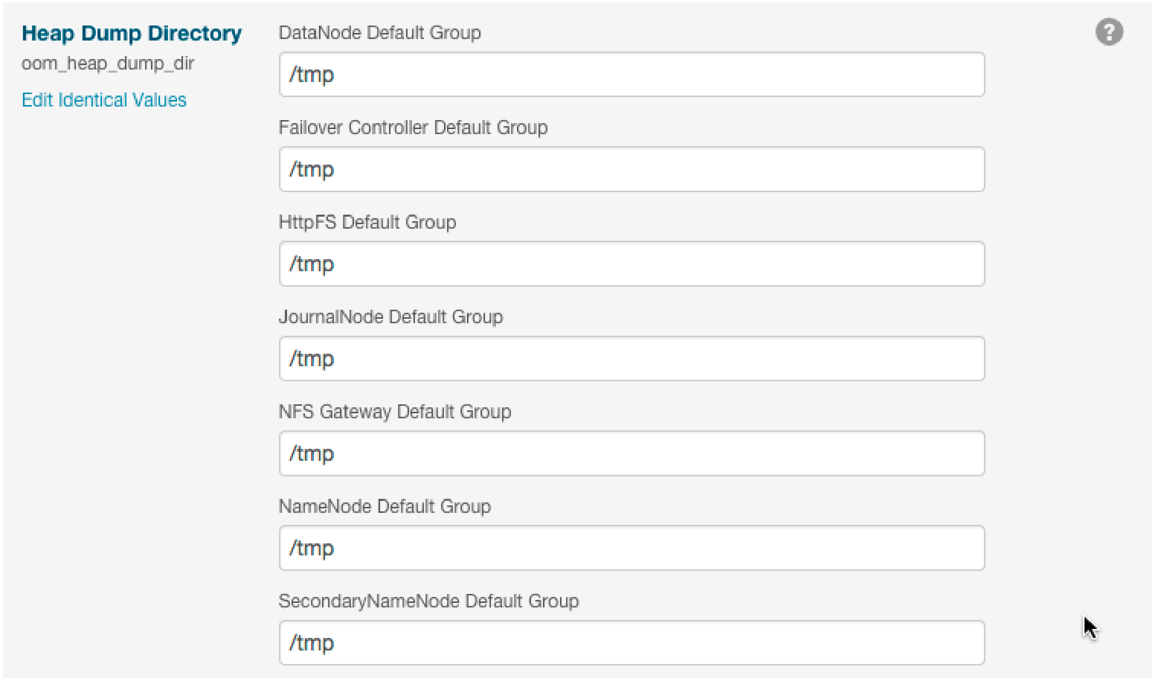


    2. Click the **... and 6 others** link to display all of the role groups:



    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:



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 32 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:

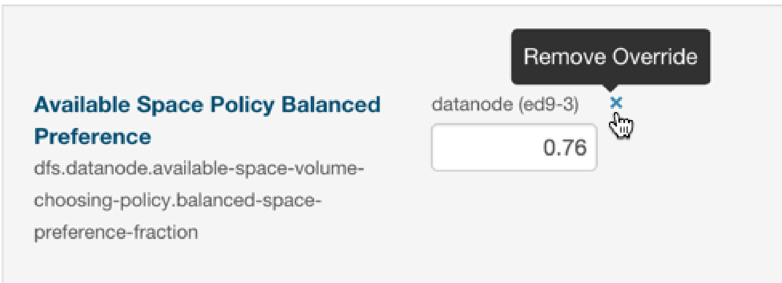
1. Go to the **Status** page for the service whose role you want to change.
2. Click the **Instances** tab.
3. Click the role instance you want to change.
4. Click the **Configuration** tab.
5. Change the configuration values as appropriate.
6. Save your changes.

You will most likely need to restart your service or role to have your configuration changes take effect. See Stale Configuration Actions on page 34.

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

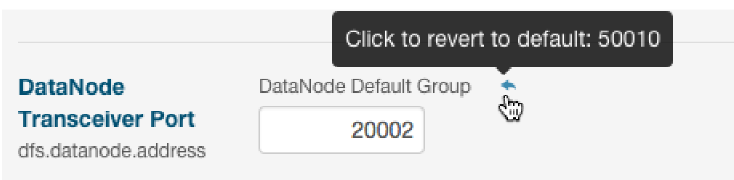
To reset a property back to its default value, click the

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.

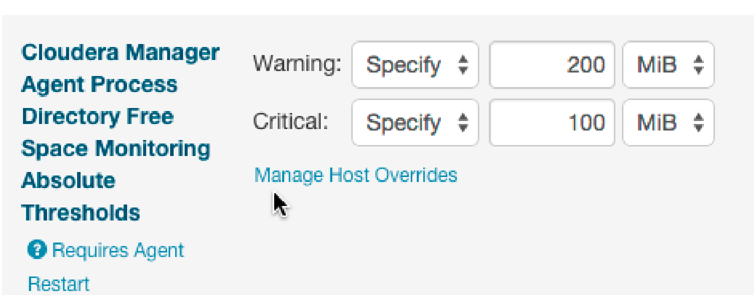


There is no mechanism for resetting to an autoconfigured value. However, you can use the configuration history and rollback feature to revert any configuration changes.

Viewing and Editing Host Overrides

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

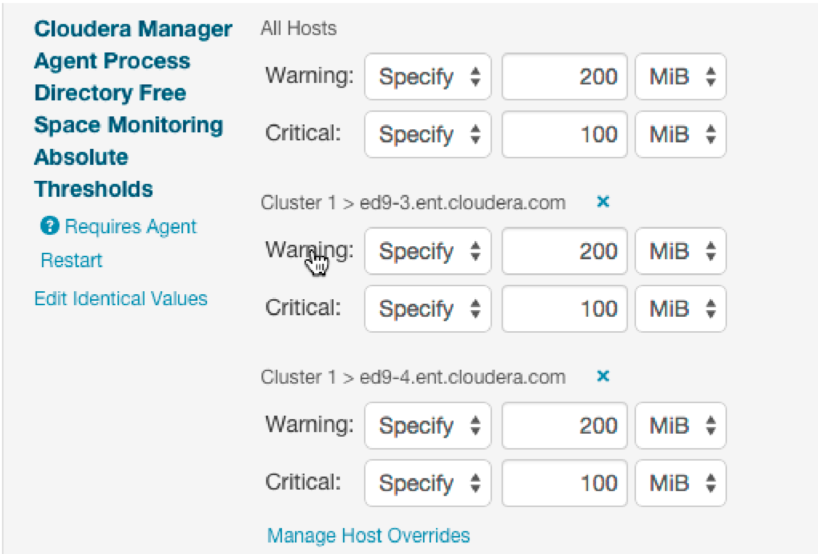
1. Click the **Hosts** tab.
2. Click the **Configuration** tab.
3. Use the Filters or Search box to locate the property that you want to override.
4. Click the **Manage Host Overrides** link.



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.

   To remove the override, click the
   
   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 46 or Starting, Stopping, and Restarting Role Instances on page 56.
2. If you see a **Finished** status, the service or role instances have restarted.
3. Go to the **Home** > **Status** tab. The service should show a Status of **Started** for all instances and a health status of **Good**.

For more information, see Stale Configurations on page 32.

Suppressing Configuration and Parameter Validation Warnings

You can suppress the warnings that Cloudera Manager issues when a configuration value is outside the recommended range or is invalid. If a warning does not apply to your deployment, you might want to suppress it. Suppressed validation warnings are still retained by Cloudera Manager, and you can unsuppress the warnings at any time. You can suppress each warning when you view it, or you can configure suppression for a specific validation before warnings occur.

Suppressing a Configuration Validation in Cloudera Manager

1. Click the **Suppress...** link to suppress the warning.

A dialog box opens where you can enter a comment about the suppression.

2. Click **Confirm**.

You can also suppress warnings from the **All Configuration Issues** screen:

1. Browse to the **Home** screen.
2. Click **Configurations** > **Configuration Issues**.
3. Locate the validation message in the list and click the **Suppress...** link.

   A dialog box opens where you can enter a comment about the suppression.

4. Click **Confirm**.

The suppressed validation warning is now hidden.

Managing Suppressed Validations

On pages where you have suppressed validations, you see a link that says **Show # Suppressed Warning(s)**. On this screen, you can:

- Click the **Show # Suppressed Warning(s)** link to show the warnings.

  Each suppressed warning displays an icon: .

- Click the **Unsuppress...** link to unsuppress the configuration validation.
- Click the **Hide Suppressed Warnings** link to re-hide the suppressed warnings.

Suppressing Configuration Validations Before They Trigger Warnings

1. Go to the service or host with the configuration validation warnings you want to suppress.
2. Click **Configuration**.
3. In the filters on the left, select **Category** > **Suppressions**.

   A list of suppression properties displays. The names of the properties begin with **Suppress Parameter Validation** or **Suppress Configuration Validator**. You can also use the **Search** function to limit the number of properties that display.

4. Select a suppression property to suppress the validation warning.
5. Click **Save Changes** to commit the changes.

Viewing a List of All Suppressed Validations

Do one of the following:

- From the **Home** page or the **Status** page of a cluster, select **Configuration** > **Suppressed Health and Configuration Issues**.
- From the **Status** page of a service, select **Configuration** > **Category** > **Suppressions** and select **Status** > **Non-default**.
- From the **Host** tab, select **Configuration** > **Category** > **Suppressions** and select **Status** > **Non-default**.

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 Using Cloudera Manager on page 13.
>
> 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 57.

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 32 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

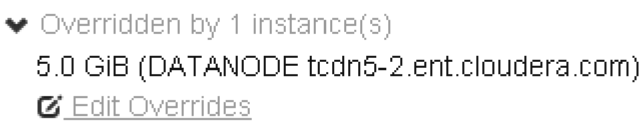
For role types that allow multiple instances, each role instance inherits its configuration properties from its associated role group. While role groups provide a convenient way to provide alternate configuration properties for selected groups of role instances, there may be situations where you want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

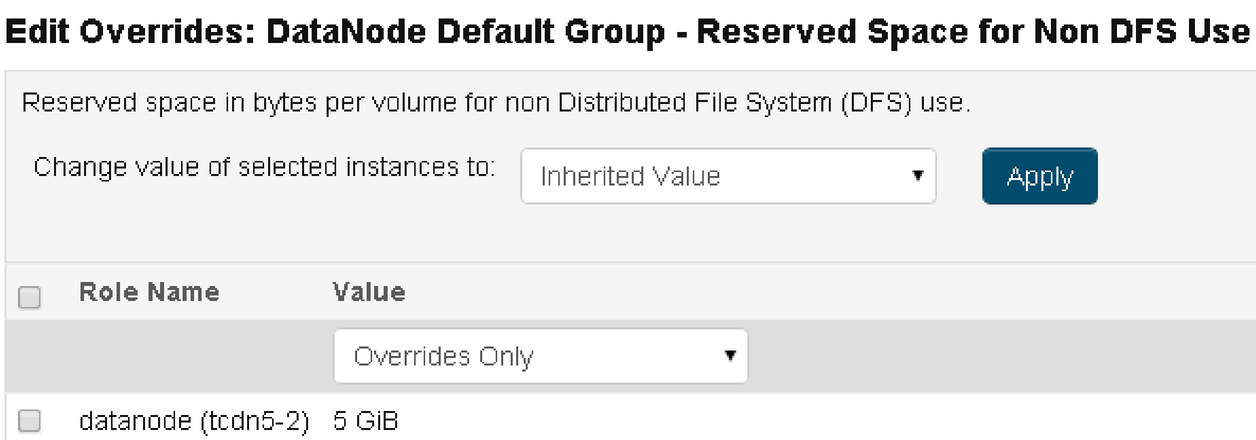
You will most likely need to restart your service or role to have your configuration changes take effect.

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.



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.



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 46 or Starting, Stopping, and Restarting Role Instances on page 56.
2. If you see a **Finished** status, the service or role instances have restarted.
3. Go to the **Home** > **Status** tab. The service should show a Status of **Started** for all instances and a health status of **Good**.

For more information, see Stale Configurations on page 32.

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, (service percentage chosen in wizard) * 20)`.

Cgroup `blkio.weight` - For each role group that supports `blkio.weight`, set to `max(100, (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 < sum(minimum) ratio = 0
        b. If RAM >= sum(ideal) ratio = 1
    d. If the host has more available memory than the total of the ideal memory for all roles assigned to the host, each role is assigned its ideal memory and autoconfiguration is finished.
    e. Cloudera Manager assigns all available host memory by setting each {role, segment} pair to the same consumption value, except in cases where that value is below the minimum memory or above the ideal memory for that pair. In that case, it is set to the minimum memory or the ideal memory as appropriate. This ensures that pairs with low ideal memory requirements are completely satisfied before pairs with higher ideal memory requirements.
5. 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).
6. 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

HDFS

For the NameNode and Secondary NameNode JVM heaps, the minimum is 50 MB and the ideal is `max(4 GB, sum_over_all(DataNode mountpoints’ available space) / 0.000008)`.

MapReduce

For the JobTracker JVM heap, the minimum is 50 MB and the ideal is `max(1 GB, round((1 GB * 2.3717181092 * ln(number of TaskTrackers in MapReduce service)) - 2.6019933306))`. If the number of TaskTrackers <= 5, the ideal is 1 GB.

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

HBase

For the memory total allowed for HBase RegionServer JVM heap, the minimum is 50 MB and the ideal is `min (31 GB ,(total RAM on region server host) * 0.64)`

YARN

For the memory total allowed for containers, the minimum is 1 GB and the ideal is `(total RAM on NodeManager host) * 0.64`.

Hue

With the exception of the Beeswax Server (only in CDH 4), Hue roles do not 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 do not 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 `(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 GB, (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 GB, (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((segment soft max (if exists) or segment max (if exists) or 2^63-1), (total RAM on role's host * 0.8 / segment scale factor * service percentage chosen in wizard * 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(cgroup.memory_limit_in_bytes_min (if exists) or 0, 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 `(total RAM on role's host * 0.8 * 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 GB, (total RAM on NodeManager host) * 0.8 * service percentage chosen in wizard)`.

Impala

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

MapReduce

- Mapper JVM heaps - the minimum is 1 and the ideal is (number of cores, including hyperthreads, on the TaskTracker host * service percentage chosen in wizard). 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 (number of cores, including hyperthreads on the TaskTracker host * service percentage chosen in wizard) / 2. 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 25.

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.

Hive

Hive on Spark rules apply only when Hive depends on YARN. The following rules are applied:

- Spark executor cores - Set to 4, 5, or 6. The value that results in the fewest "wasted" cores across the cluster is used, where the number of cores wasted per host is the remainder of `yarn.nodemanager.resource.cpu-vcores / spark.executor.cores`. In case of a tie, use the larger value of Spark executor cores. If no host on the cluster has 4 or more cores, then sets the value to the smallest value of `yarn.nodemanager.resource.cpu-vcores` on the cluster.
- Spark executor memory - 85% of Spark executor memory allocated to `spark.executor.memory` and 15% allocated to `spark.yarn.executor.memoryOverhead`. The total memory is the YARN container memory split evenly between the maximum number of executors that can run on a host. This is `yarn.nodemanager.resource.memory-mb / floor(yarn.nodemanager.resource.cpu-vcores / spark.executor.cores)`. When the memory or vcores vary across hosts in the cluster, choose the smallest calculated value for Spark executor memory.
- Spark driver memory - 90% of Spark driver memory allocated to `spark.driver.memory` and 10% allocated to `spark.yarn.driver.memoryOverhead`. The total memory is based on the lowest value of `yarn.nodemanager.resource.memory-mb` across the cluster.
- Total memory is:
  - 12 GB when `yarn.nodemanager.resource.memory-mb` > 50 GB.
  - 4 GB when `yarn.nodemanager.resource.memory-mb` < 50 GB && >= 12 GB
  - 1 GB when `yarn.nodemanager.resource.memory-mb` < 12 GB
  - 256 MB when `yarn.nodemanager.resource.memory-mb` < 1 GB.

The rules apply in the cases described in General Rules on page 27 and on upgrade from Cloudera Manager 5.6.x and lower to Cloudera Manager 5.7.x and higher.

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 higher 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)`.
- `mapred.job.tracker.handler.count` - For each JobTracker, set to `max(10, ln(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(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,sum_over_all(NodeManager number of cores, including hyperthreads) / 2)`.
- `yarn.resourcemanager.nodemanagers.heartbeat-interval-ms` - For each NodeManager, set to `max(100, 10 * (number of NodeManagers in this YARN service))`.
- `yarn.scheduler.maximum-allocation-vcores` - For each ResourceManager, set to `max_over_all(NodeManager number of vcores (determined via yarn.nodemanager.resource.cpu-vcores for that NodeManager, which is configured separately))`.
- `yarn.scheduler.maximum-allocation-mb` - For each ResourceManager, set to `max_over_all(NodeManager amount of RAM (determined via yarn.nodemanager.resource.memory-mb 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(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.

Server and client configurations have separate configuration snippets. In general after changing a server configuration snippet you must restart the server, and after changing a client configuration snippet you must redeploy 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 indicates the actions you must perform to resolve stale configurations.

Configuration Snippet Types and Syntax

Configuration

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 `.xml` or `.conf`.

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**; configurations you enter here are inserted verbatim into the `hive-site.xml` file associated with the HiveServer2 role group.

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 `mapred-site.xml` shows the configuration snippets that have `mapred-site.xml` in their name.

Some configuration snippet descriptions include the phrase *for this role only*. These configurations are stored in memory, and only inserted to the configuration when running an application from Cloudera Manager. Otherwise, the configuration changes are added to the configuration file on disk, and are used when running the application both from Cloudera Manager and from the command line.

Syntax:

```
<property>
  <name>property_name</name>
  <value>property_value</value>
</property>
```

For example, to specify a MySQL connector library, put this property definition in that configuration snippet:

```
<property>
  <name>hive.aux.jars.path</name>
  <value>file:///usr/share/java/mysql-connector-java.jar</value>
</property>
```

Environment

Specify key-value pairs for a service, role, or client that are inserted into the respective environment.

One example of using an environment configuration snippet is to add a JAR to a classpath. Place JARs in a custom location such as `/opt/myjars` and extend the classpath using 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.

Do not place JARs inside locations such as `/opt/cloudera` or `/usr/lib/{hadoop*,hbase*,hive*}` that are managed by Cloudera because they are overwritten at upgrades.

Syntax:

`key=value`

For example, to add JDBC connectors to a Hive gateway classpath, add

```
AUX_CLASSPATH=/usr/share/java/mysql-connector-java.jar:\
/usr/share/java/oracle-connector-java.jar
```

or

```
AUX_CLASSPATH=/usr/share/java/*
```

to **Gateway Client Advanced Configuration Snippet for hive-env.sh**.

Logging

Set log4j properties in a `log4j.properties` file.

Syntax:

```
key1=value1
key2=value2
```

For example:

```
log4j.rootCategory=INFO, console max.log.file.size=200MB
max.log.file.backup.index=10
```

Metrics

Set properties to configure Hadoop metrics in a `hadoop-metrics.properties` or `hadoop-metrics2.properties` file.

Syntax:

```
key1=value1
key2=value2
```

For example:

```
*.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
```

Whitelists and blacklists

Specify a list of host addresses that are allowed or disallowed from accessing a service.

Syntax:

```
host1.domain1 host2.domain2
```

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. If the snippet is an XML file, you have the option to use a snippet editor (the default) or an XML text field:

- Snippet editor

1. Click ✚ to add a property. Enter the property name, value, and optional description. To indicate that the property value cannot be overridden by another , select the **Final** checkbox.

- XML text field - Enter the property name, value, and optional description in as XML elements.

```
<property>
  <name>name</name>
  <value>property_value</value>
  <final>final_value</final>
</property>
```

To indicate that the property value cannot be overridden, specify `<final>true</final>`.

To switch between the editor and text field, click the **View Editor** and **View XML** links at the top right of the snippet row.

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** > **Status** tab, click a cluster name.
     2. Select **Configuration** > **Advanced Configuration Snippets**.
   - **all clusters**
     1. Select **Configuration** > **Advanced Configuration Snippets**.
2. Specify the snippet properties. If the snippet is an XML file, you have the option to use a snippet editor (the default) or an XML text field:
   - Snippet editor

     1. Click + to add a property. Enter the property name, value, and optional description. To indicate that the property value cannot be overridden by another , select the **Final** checkbox.
   - XML text field - Enter the property name, value, and optional description in as XML elements.

```
<property>
  <name>name</name>
  <value>property_value</value>
  <final>final_value</final>
</property>
```

To indicate that the property value cannot be overridden, specify `<final>true</final>`.

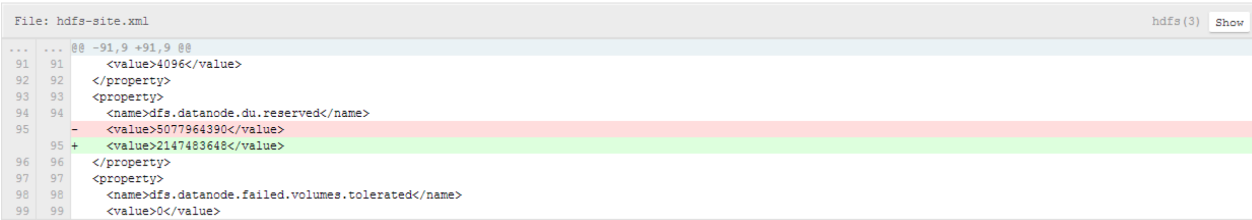
To switch between the editor and text field, click the **View Editor** and **View XML** links at the top right of the snippet row.

3. Click **Save Changes** to commit the changes.
4. Restart the service or role or redeploy client configurations as indicated.

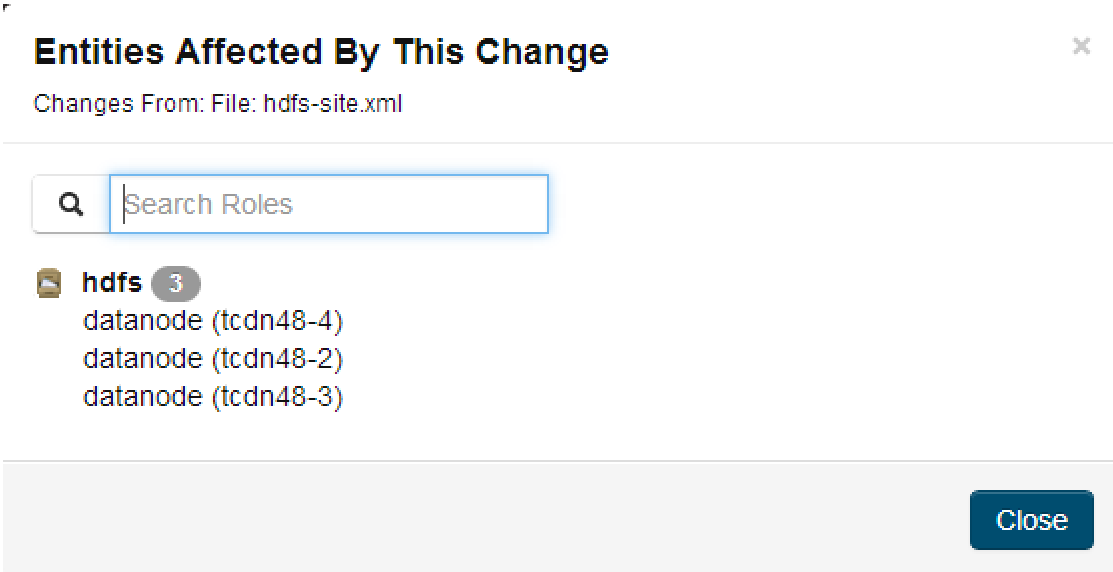
Stale Configurations

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:



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:



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.

- **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**.

Stale Configuration Actions

The Stale Configurations page displays action buttons. 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 Stale Services** if *one* of the roles listed on the page need to be restarted.

- **Refresh Stale Services** - Refreshes stale services.
- **Restart Stale Services** - Restarts stale services.
- **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** scope 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 32 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	1. On the **Home** > **Status** tab, 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 `wget` or `curl`.
Service	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** > **Status** tab, 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 583.

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 57 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, however each cluster can only be associated with a single Cloudera Manager Server or Cloudera Manager HA pair. 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** > **Status** tab 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 10.

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

Action	Procedure
	1. On the **Home** > **Status** tab, 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.

View a video about How to Add a Cluster to Cloudera Manager.

Deleting a Cluster

1. Stop the cluster.
2. On the **Home** > **Status** tab, 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** > **Status** tab, 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** > **Status** tab, click

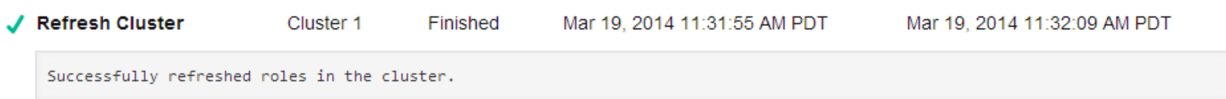
   to the right of the cluster name and select **Stop**.
2. Click **Stop** in the confirmation screen. The **Command Details** window shows the progress of stopping services.

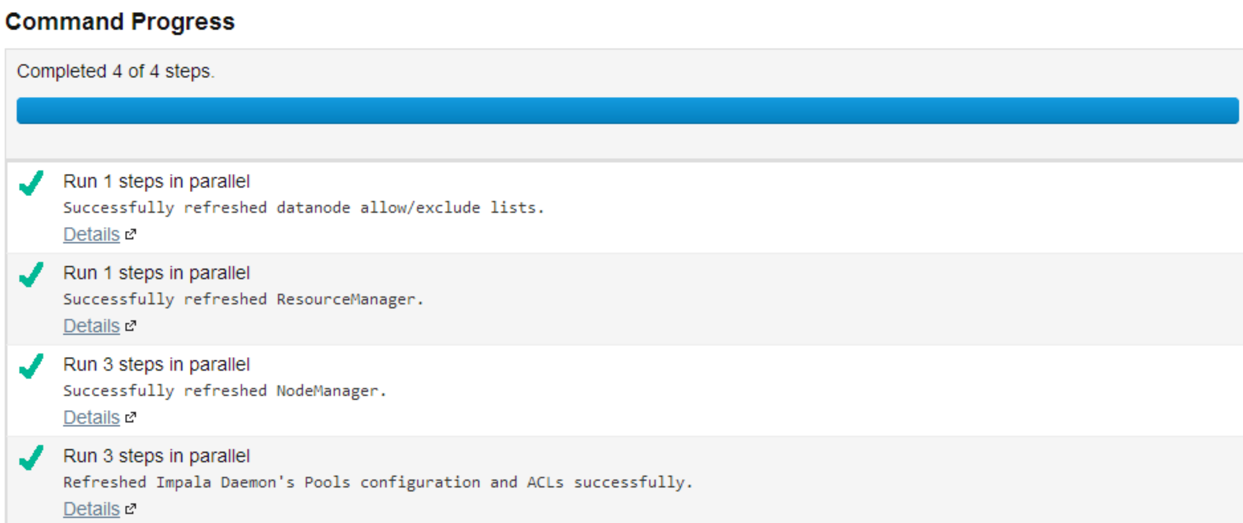
   When **All services successfully stopped** appears, the task is complete and you can close the **Command Details** window.

> **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:





To refresh a cluster, in the **Home** > **Status** tab, click

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

Restarting a Cluster

1. On the **Home** > **Status** tab, 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** > **Status** tab, 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 56).
2. Removing all roles from the host (except for the Cloudera Manager management roles). See Deleting Role Instances on page 57.
3. Deleting the host from the cluster (see Deleting Hosts on page 71), 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 64).
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 55 and Host Templates on page 67.

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 91.

> **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 10.

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 and must be installed individually before being adding the service:

Service	Installation Documentation
Accumulo	Apache Accumulo Documentation
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. Select 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 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 box 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 box.

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** > **Status** tab, 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** > **Status** tab. For more information, see Cluster-Wide Configuration on page 40

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 64 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 43 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:

   `systemd`-based Operating Systems:

```
sudo systemctl restart cloudera-scm-server
```

   `init`-based Operating Systems:

```
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. If their software is not available as a parcel, then you must install their software *before* adding the CSD file. Follow the instructions from the ISV for installing the software. 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** > **Status** tab, 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

> **Note:** It is not required that the Cloudera Manager server host be part of a managed cluster and have an agent installed. Although you initially copy the CSD file to the Cloudera Manager server, the Parcel for the add-on service will not be installed on the Cloudera Manager Server host unless the host is managed by Cloudera Manager.

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 44.

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 or click **Hosts** and click the **Parcels** tab.
     2. Click the **Configuration** 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 41.

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 38.

Starting a Service on All Hosts

1. On the **Home** > **Status** tab, 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 49.

Stopping a Service on All Hosts

1. On the **Home** > **Status** tab, 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** > **Status** tab, 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 583.

Rolling restart allows you to conditionally restart the role instances of the following services to update software or use a new configuration:

- Flume
- HBase
- HDFS
- Kafka
- Key Trustee KMS
- Key Trustee Server
- MapReduce
- Oozie
- Sentry
- YARN
- ZooKeeper

If the service is not running, rolling restart is not available for that service. You can specify 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, to avoid having your NameNode (and thus the cluster) be unavailable during the restart, Cloudera Manager forces a failover to the standby NameNode.

MapReduce (MRv1) JobTracker High Availability on page 405 and YARN (MRv2) ResourceManager High Availability on page 396 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. Cloudera recommends 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.
>   - Another option to increase the speed of a rolling restart of the HBase service is to set the **Skip Region Reload During Rolling Restart** property to `true`. This setting can cause regions to be moved around multiple times, which can degrade HBase client performance.
> - **MapReduce** - If you restart the JobTracker, all current jobs will fail.
> - **YARN** - If you restart ResourceManager and ResourceManager HA is enabled, current jobs continue running: they do not restart or fail. ResourceManager 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 372 for instructions. You do not need to enable automatic failover for rolling restart to work, though you can enable it if you want. 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 **Restart** 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 45.
2. On the **Home** > **Status** tab, 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** > **Status** tab, 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 set 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.

Exposing Hadoop Metrics to Graphite

Core Hadoop services and HBase support the writing of their metrics to Graphite, a real-time graphing system.

HDFS, YARN, and HBase support the Metrics2 framework; MapReduce1 and HBase support the Metrics framework. See the Cloudera blog post, What is Hadoop Metrics2?

Configure Hadoop Metrics for Graphite Using Cloudera Manager

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

1. Go to the **Home** page by clicking the Cloudera Manager logo.
2. Click **Configuration** > **Advanced Configuration Snippets**.
3. Search on the term *Metrics*.
4. To configure HDFS, YARN, or HBase, use **Hadoop Metrics2 Advanced Configuration Snippet (Safety Valve)**. For MapReduce1 (or HBase), use **Hadoop Metrics Advanced Configuration Snippet (Safety Valve)**.
5. Click **Edit Individual Values** to see the supported daemons and their default groups.
6. Configure each default group with a metrics class, sampling period, and Graphite server. See the tables below.
7. To add optional parameters for socket connection retry, modify this example as necessary:

```
*.sink.graphite.retry_socket_interval=60000 #in milliseconds
*.sink.graphite.socket_connection_retries=10 #Set it to 0 if you do not want it to be
retried
```

8. Click **Save Changes**.
9. **Restart** the cluster or service depending on the scope of your changes.

Graphite Configuration Settings Per Daemon

Table 1: Hadoop Metrics2 Graphite Configuration

Service	Daemon Default Group	Graphite Configuration Settings

HBase	Master and RegionServer	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`hbase.sink.graphite.server_host=<hostname>`
		`hbase.sink.graphite.server_port=<port>`
		`hbase.sink.graphite.metrics_prefix=<prefix>`

HDFS	DataNode	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`datanode.sink.graphite.server_host=<hostname>`
		`datanode.sink.graphite.server_port=<port>`
		`datanode.sink.graphite.metrics_prefix=<prefix>`

	NameNode	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`namenode.sink.graphite.server_host=<hostname>`
		`namenode.sink.graphite.server_port=<port>`
		`namenode.sink.graphite.metrics_prefix=<prefix>`

	SecondaryNameNode	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`secondarynamenode.sink.graphite.server_host=<hostname>`
		`secondarynamenode.sink.graphite.server_port=<port>`
		`secondarynamenode.sink.graphite.metrics_prefix=<prefix>`

YARN	NodeManager	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`nodemanager.sink.graphite.server_host=<hostname>`
		`nodemanager.sink.graphite.server_port=<port>`
		`nodemanager.sink.graphite.metrics_prefix=<prefix>`

	ResourceManager	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`resourcemanager.sink.graphite.server_host=<hostname>`
		`resourcemanager.sink.graphite.server_port=<port>`
		`resourcemanager.sink.graphite.metrics_prefix=<prefix>`

	JobHistory Server	`*.sink.graphite.class=org.apache.hadoop.metrics2.sink.GraphiteSink`
		`*.period=10`
		`jobhistoryserver.sink.graphite.server_host=<hostname>`
		`jobhistoryserver.sink.graphite.server_port=<port>`
		`jobhistoryserver.sink.graphite.metrics_prefix=<prefix>`

> **Note:** To use metrics, set values for each *context*. For example, for MapReduce1, add values for both the **JobTracker Default Group** and the **TaskTracker Default Group**.

Table 2: Hadoop Metrics Graphite Configuration

Service	Daemon Default Group	Graphite Configuration Settings
HBase	Master	`hbase.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
	RegionServer	

Service	Daemon Default Group	Graphite Configuration Settings

		`hbase.period=10`
		`hbase.servers=<graphite hostname>:<port>`
		
		`jvm.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`jvm.period=10`
		`jvm.servers=<graphite hostname>:<port>`
		
		`rpc.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`rpc.period=10`
		`rpc.servers=<graphite hostname>:<port>`

MapReduce1	JobTracker
	TaskTracker
		`dfs.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`dfs.period=10`
		`dfs.servers=<graphite hostname>:<port>`
		
		`mapred.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`mapred.period=10`
		`mapred.servers=<graphite hostname>:<port>`
		
		`jvm.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`jvm.period=10`
		`jvm.servers=<graphite hostname>:<port>`
		
		`rpc.class=org.apache.hadoop.metrics.graphite.GraphiteContext`
		
		`rpc.period=10`
		`rpc.servers=<graphite hostname>:<port>`

Exposing Hadoop Metrics to Ganglia

Core Hadoop services and HBase support the writing of their metrics to Ganglia, a data representation and visualization tool.

HDFS, YARN, and HBase support the Metrics2 framework; MapReduce1 and HBase support the Metrics framework. See the Cloudera blog post, What is Hadoop Metrics2?

Configure Hadoop Metrics for Ganglia Using Cloudera Manager

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

1. Go to the **Home** page by clicking the Cloudera Manager logo.
2. Click **Configuration** > **Advanced Configuration Snippets**.
3. Search on the term *Metrics*.
4. To configure HDFS, YARN, or HBase, use **Hadoop Metrics2 Advanced Configuration Snippet (Safety Valve)**. For MapReduce1 (or HBase), use **Hadoop Metrics Advanced Configuration Snippet (Safety Valve)**.
5. Click **Edit Individual Values** to see the supported daemons and their default groups.
6. Configure each default group with a metrics class, sampling period, and Ganglia server. See the tables below.
7. To add optional parameters for socket connection retry, modify this example as necessary:

```
*.sink.ganglia.retry_socket_interval=60000 #in milliseconds
*.sink.ganglia.socket_connection_retries=10 #Set it to 0 if you do not want it to be
retried
```

8. To define a filter, which is recommended for preventing YARN metrics from overwhelming the Ganglia server, do so *on the sink side*. For example:

```
*.source.filter.class=org.apache.hadoop.metrics2.filter.GlobFilter
*.record.filter.class=${*.source.filter.class}
*.metric.filter.class=${*.source.filter.class}
nodemanager.sink.ganglia.record.filter.exclude=ContainerResource*
```

9. Click **Save Changes**.
10. **Restart** the Cluster or Service depending on the scope of your changes.

Ganglia Configuration Settings Per Daemon

Table 3: Hadoop Metrics2 Ganglia Configuration

Service	Daemon Default Group	Ganglia Configuration Settings
HBase	Master and RegionServer	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 hbase.sink.ganglia.servers=<hostname>:<port>
HDFS	DataNode	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 datanode.sink.ganglia.servers=<hostname>:<port>
	NameNode	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 namenode.sink.ganglia.servers=<hostname>:<port>
	SecondaryNameNode	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 secondarynamenode.sink.ganglia.servers=<hostname>:<port>
YARN	NodeManager	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 nodemanager.sink.ganglia.servers=<hostname>:<port>
	ResourceManager	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 resourcemanager.sink.ganglia.servers=<hostname>:<port>
	JobHistory Server	*.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31 *.period=10 jobhistoryserver.sink.ganglia.servers=<hostname>:<port>

Note: To use metrics, set values for each *context*. For example, for MapReduce1, add values for both the **JobTracker Default Group** and the **TaskTracker Default Group**.

Table 4: Hadoop Metrics Ganglia Configuration

Service	Daemon Default Group	Ganglia Configuration Settings
HBase	Master	hbase.class=org.apache.hadoop.metrics.ganglia.GangliaContext31 hbase.period=10 hbase.servers=<hostname>:<port> jvm.class=org.apache.hadoop.metrics.ganglia.GangliaContext31 jvm.period=10 jvm.servers=<hostname>:<port> rpc.class=org.apache.hadoop.metrics.ganglia.GangliaContext31 rpc.period=10 rpc.servers=<hostname>:<port>
	RegionServer	
MapReduce1	JobTracker	dfs.class=org.apache.hadoop.metrics.ganglia.GangliaContext dfs.period=10 dfs.servers=<hostname>:<port> mapred.class=org.apache.hadoop.metrics.ganglia.GangliaContext mapred.period=10 mapred.servers=<hostname>:<port> jvm.class=org.apache.hadoop.metrics.ganglia.GangliaContext jvm.period=10 jvm.servers=<hostname>:<port> rpc.class=org.apache.hadoop.metrics.ganglia.GangliaContext rpc.period=10 rpc.servers=<hostname>:<port>
	TaskTracker	

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 10.

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, Kafka, 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.

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 box 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 box.

   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 58 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 46.

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. If you attempt to decommission a DataNode or a host with a DataNode in such situations, the DataNode will be decommissioned, but the decommission process will not complete. You will have to abort the decommission and recommission the DataNode.

A role will be decommissioned if its host is decommissioned. See Decommissioning and Recommissioning Hosts on page 69 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 69.
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. A Decommission Command pop-up displays that shows each step or decommission command as it is run. In the Details area, click to see the subcommands that are run. 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.

   You can abort the decommission process by clicking the **Abort** button, but you must recommission and restart the role.

   The Commission State facet in the Filters list displays Decommissioning while decommissioning is in progress, and Decommissioned when the decommissioning process has finished. When the process is complete, a ✓ is added in front of Decommission Command.

Recommissioning Role Instances

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

1. Click the service 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** to start the process. A Recommission Command pop-up displays that shows each step or recommission command as it is run. When the process is complete, a ✓ is added in front of Recommission Command.
5. Restart the role instance.

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 Using Cloudera Manager on page 13.

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	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	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. 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	1. Select the role instance(s) to move. 2. Select **Actions for Selected** > **Move To Different Role Group...**.

Action	Procedure
	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.

The Hosts screen has the following tabs:

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.
- Click › to the left of the number of roles to list all the role instances running on that host.
- Filter the hosts list by entering search terms (hostname, IP address, or role) in the search box separated by commas or spaces. Use quotes for exact matches (for example, strings that contain spaces, such as a role name) and brackets to search for ranges. Hosts that match any of the search terms are displayed. For example:

```
hostname[1-3], hostname8 hostname9, "hostname.example.com"
hostname.example.com “HDFS DataNode”
```

- You can also search for hosts by selecting a value from the facets in the **Filters** section at the left of the page.
- If the Configuring Agent Heartbeat and Health Status Options on page 569 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 a host's Last Heartbeat facet is computed as follows:

  - $v < x * y$ = Good
  - $v >= x * y$ and $<= x * z$ = Concerning
  - $v >= x * z$ = Bad

Viewing the Hosts in a Cluster

Do one of the following:

- Select **Clusters** > ***Cluster name*** > **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 60.

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 60) 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 Using Cloudera Manager on page 13.

The Roles and Disks Overview Tabs

Role Assignments

You can view the assignment of roles to hosts as follows:

1. Click the **Roles** tab.
2. Click a cluster name or **All Clusters**.

Disks Overview

Click the **Disks Overview** tab 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"` and click **Filter**. Click a histogram to drill down and identify outliers. Mouse over the graph and click to display additional information about the chart.

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 55.

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 Using Cloudera Manager on page 13.

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 Lifecycle and Security Auditing 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 and select **All Hosts**.
2. Click the **Inspect All Hosts** button. 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.

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 cluster using the Add Hosts wizard, which installs the Oracle JDK, CDH, and Cloudera Manager Agent software. After the software is installed and the Cloudera Manager Agent is started, the Agent connects to the Cloudera Manager Server and you can use the Cloudera Manager Admin Console to manage and monitor CDH on the new host.

> **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 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:**
>
> - All hosts in a single cluster must be running the same version of CDH.
> - When you add a new host, 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 to install, and you can choose a custom repository to ensure that the version you install matches the version on the other hosts.
> - If you are managing multiple clusters, select the version of CDH that matches the version in use on the cluster where you plan to add the new host.
> - When you add a new host, the following occurs:
>   - YARN `topology.map` is updated to include the new host
>   - Any service that includes `topology.map` in its configuration—Flume, Hive, Hue, Oozie, Solr, Spark, Sqoop 2, YARN—is marked stale
>
>   At a convenient point after adding the host you should restart the stale services to pick up the new configuration.

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 temporarily puts the existing hosts in an unmanageable state; they are still configured to use TLS and so cannot communicate with the Cloudera Manager Server. Roles on these hosts continue to operate normally.

1. From the **Administration** tab, select **Settings**.
2. Select the **Security** category.
3. Disable all levels of TLS that are currently enabled by clearing 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. Log in to the Cloudera Manager Server host.
6. Restart the Cloudera Manager Server with the following command:

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

The changes take effect after the restart.

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 67 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 55 for information about adding roles for a specific service. See Host Templates on page 67 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 TLS/SSL for CDH Components

If you have previously enabled TLS/SSL on your cluster, and you plan to start these roles on this new host, make sure you install a new host certificate to be configured from the same path and naming convention as the rest of your hosts. Since the new host and the roles configured on it are inheriting their configuration from the previous host, ensure that the keystore or truststore passwords and locations are the same on the new host. For instructions on configuring TLS/SSL, see Configuring TLS/SSL Encryption for CDH Services.

Enable Kerberos

If you have previously enabled Kerberos on your cluster:

1. Install the packages required to `kinit` on the new host (see the list in Before you Begin Using the Wizard).
2. If you have set up Cloudera Manager to manage `krb5.conf`, it will automatically deploy the file on the new host. Note that Cloudera Manager will deploy `krb5.conf` only if you use the Kerberos wizard. If you have used the API, you will need to manually perform the commands that the wizard calls.

   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 - Installation Using Cloudera Manager Parcels or Packages.
2. After installation is complete, start the Cloudera Manager Agent. For instructions, see Starting, Stopping, and Restarting Cloudera Manager Agents on page 568.
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 TLS/SSL on your cluster, and you plan to start these roles on this new host, make sure you install a new host certificate to be configured from the same path and naming convention as the rest of your hosts. Since the new host and the roles configured on it are inheriting their configuration from the previous host, ensure that the keystore or truststore passwords and locations are the same on the new host. For instructions on configuring TLS/SSL, see Configuring TLS/SSL Encryption for CDH Services.

6. If you have previously enabled Kerberos on your cluster:

   1. Install the packages required to `kinit` on the new host (see the list in Before you Begin Using the Wizard).
   2. If you have set up Cloudera Manager to manage `krb5.conf`, it will automatically deploy the file on the new host. Note that Cloudera Manager will deploy `krb5.conf` only if you use the Kerberos wizard. If you have used the API, you will need to manually perform the commands that the wizard calls.

      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 they 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.

Decommissioning and Recommissioning Hosts

Decommissioning a host decommissions and stops all roles on the host without requiring you to individually decommission the roles on each service. 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.

After 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. If you attempt to decommission a DataNode or a host with a DataNode in such situations, the DataNode will be decommissioned, but the decommission process will not complete. You will have to abort the decommission and recommission the DataNode.

To decommission hosts:

1. If the host has a DataNode, perform the steps in Tuning HDFS Prior to Decommissioning DataNodes on page 69.
2. Click the **Hosts** tab.
3. Select the checkboxes next to one or more hosts.
4. Select **Actions for Selected** > **Hosts 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 Decommission Command pop-up displays that shows each step or decommission command as it is run, service by service. In the Details area, click › to see the subcommands that are run for decommissioning a given service. Depending on the service, 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.

   You can abort the decommission process by clicking the **Abort** button, but you must recommission and restart each role that has been decommissioned.

   The Commission State facet in the Filters lists displays Decommissioning while decommissioning is in progress, and Decommissioned when the decommissioning process has finished. When the process is complete, a ✓ is added in front of Decommission Command.

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 from the DataNode in small batches. If 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 Using Cloudera Manager on page 13.
   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 Using Cloudera Manager on page 13.
   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 Using Cloudera Manager on page 13.
   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** and **Confirm**. A Recommission Command pop-up displays that shows each step or recommission command as it is run. When the process is complete, a ✓ is added in front of Recommission Command. The host and roles are marked as commissioned, but the roles themselves are not restarted.

Restarting All The Roles on a 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 all roles.
3. Select **Actions for Selected** > **Start Roles on Hosts**.

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 568.
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.

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, the roles associated with that service are entered into effective maintenance mode, and remain in effective maintenance mode until the service exits maintenance mode. You cannot remove them from maintenance mode individually.

Alternatively, an entity that is in effective maintenance mode can be put into explicit maintenance mode. In this case, the entity remains 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 remains in maintenance mode. You 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. Go to the service page in Cloudera Manager.

2. Click **Actions** > **Enter Maintenance Mode**.
3. 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.

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.

Cloudera Manager Configuration Properties

Refer to the links below for a list of available CDH properties for a given version of CDH when managed by Cloudera Manager 5.9.x.

For information on managing configuration settings, see Modifying Configuration Properties Using Cloudera Manager on page 13 and Viewing and Reverting Configuration Changes on page 35.

- Cloudera Enterprise 5.9.x CDH Properties
- Cloudera Enterprise 5.8.x CDH Properties
- Cloudera Enterprise 5.7.x CDH Properties
- Cloudera Enterprise 5.6.x CDH Properties
- Cloudera Enterprise 5.5.x CDH Properties
- Cloudera Enterprise 5.4.x CDH Properties
- Cloudera Enterprise 5.3.x CDH Properties
- Cloudera Enterprise 5.2.x CDH Properties
- Cloudera Manager 5.1.x CDH Properties
- Cloudera Manager 5.0.x CDH Properties

Managing CDH Using the Command Line

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

For installation and upgrade instructions, see the #unique_139 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 Using the Command Line on page 75
  - Stopping CDH Services Using the Command Line on page 80
- Using Apache Avro Data Files with CDH
- Flume configuration
- HBase configuration:
  - Configuration Settings for HBase
  - HBase Replication on page 505
  - 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 Apache Parquet Data Files with CDH
- Snappy: Snappy Compression
- Spark configuration:
  - Managing Spark Standalone Using the Command Line on page 252
  - Running Spark Applications
- Sqoop configuration:
  - Setting HADOOP_MAPRED_HOME for Sqoop
  - Configuring Sqoop 2
- ZooKeeper: Maintaining a ZooKeeper Server

Starting CDH Services Using the Command Line

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 385; 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 372
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

Order	Service	Comments	For instructions and more information
5	HBase		Starting and Stopping HBase on page 104; 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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

`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 hosts 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	`$ sudo update-rc.d hadoop-hdfs-namenode defaults`
On the JobTracker	`$ sudo update-rc.d hadoop-0.20-mapreduce-jobtracker defaults`
On the Secondary NameNode (if used)	`$ sudo update-rc.d hadoop-hdfs-secondarynamenode defaults`
On each TaskTracker	`$ sudo update-rc.d hadoop-0.20-mapreduce-tasktracker defaults`
On each DataNode	`$ sudo update-rc.d hadoop-hdfs-datanode defaults`

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

Important:

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

The `chkconfig` commands to use are:

Where	Command
On the NameNode	`$ sudo chkconfig hadoop-hdfs-namenode on`
On the ResourceManager	`$ sudo chkconfig hadoop-yarn-resourcemanager on`
On the Secondary NameNode (if used)	`$ sudo chkconfig hadoop-hdfs-secondarynamenode on`
On each NodeManager	`$ sudo chkconfig hadoop-yarn-nodemanager on`
On each DataNode	`$ sudo chkconfig hadoop-hdfs-datanode on`
On the MapReduce JobHistory host	`$ sudo chkconfig hadoop-mapreduce-historyserver on`

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

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

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	`$ sudo chkconfig hue on`
Oozie	Oozie server	`$ sudo chkconfig oozie on`
HBase	HBase master	`$ sudo chkconfig hbase-master on`
	On each HBase RegionServer	`$ sudo chkconfig hbase-regionserver on`
Hive Metastore	Hive Metastore server	`$ sudo chkconfig hive-metastore  on`
HiveServer2	HiveServer2	`$ sudo chkconfig hive-server2 on`

Component	Server	Command
Zookeeper	Zookeeper server	`$ sudo chkconfig zookeeper-server on`
HttpFS	HttpFS server	`$ sudo chkconfig hadoop-httpfs on`

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

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

Starting and Stopping HBase Using the Command Line

When starting and stopping CDH services, order is important. See Starting CDH Services Using the Command Line on page 75 and Stopping CDH Services Using the Command Line on page 80 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 5.9.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 HMasters, 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 HMasters, 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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

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		`sudo service hue stop`

2	Impala		`sudo service impala-server stop`
			`sudo service impala-catalog stop`
			`sudo service impala-state-store stop`

3	Oozie		`sudo service oozie stop`

4	Hive	Exit the Hive console and ensure no Hive scripts are running. Stop the Hive server, HCatalog, and metastore daemon on each client.	`sudo service hiveserver2 stop`
			`sudo service hive-webhcat-server stop`
			`sudo service hive-metastore stop`

5	Flume 1.x	There is no Flume master.	`sudo service flume-ng-agent stop`
6	Sqoop 1		`sudo service sqoop-metastore stop`
6	Sqoop 2		`sudo service sqoop2-server stop`
7	Lily HBase Indexer (Solr/HBase Indexer)		`sudo service hbase-solr-indexer stop`

10	Spark		`sudo service spark-worker stop`
			`sudo service spark-history-server stop`
			`sudo service spark-master stop`

8	Sentry	Only present on a secure configuration.	`sudo service sentry-store stop`
9	Solr Search		`sudo service solr-server stop`

10	HBase	Stop the Thrift server and clients, followed by RegionServers and finally the Master.	`sudo service hbase-thrift stop`
			`sudo service hbase-rest stop`
			`sudo service hbase-regionserver stop`
			`sudo service hbase-master stop`

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.	`sudo service hadoop-0.20-mapreduce-jobtrackerha stop`
			For Non-HA setup:
			`sudo service hadoop-0.20-mapreduce-jobtracker stop`
			For all types of MRv1 setups:
			`sudo service hadoop-0.20-mapreduce-tasktracker stop`

12	YARN	Stop the JobHistory server, followed by the ResourceManager and each of the NodeManagers.	`$ sudo service hadoop-mapreduce-historyserver stop`
			`$ sudo service hadoop-yarn-resourcemanager stop`
			`$ sudo service hadoop-yarn-nodemanager stop`

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.
			`sudo service hadoop-httpfs stop`
			`sudo service hadoop-hdfs-nfs3 stop`
			`sudo service hadoop-hdfs-secondarynamenode stop`
			`sudo service hadoop-hdfs-namenode stop`
			`sudo service hadoop-hdfs-journalnode stop`
			`sudo service hadoop-hdfs-datanode stop`

14	KMS (Key Management Server)	Only present if HDFS at rest encryption is enabled	`sudo service hadoop-kms-server stop`
15	ZooKeeper		`sudo service zookeeper-server stop`

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 Cluster Data Using DistCp

The distributed copy command, `distcp`, is a general utility for copying large data sets between distributed filesystems within and across clusters. You can also use `distcp` to copy data to and from an Amazon S3 bucket. The `distcp` command submits a regular MapReduce job that performs a file-by-file copy.

To see the `distcp` command options, run the built-in help:

```
$ hadoop distcp
```

Important:

- The `distcp` command is not supported when copying data between encrypted clusters.
- Do not run `distcp` as the `hdfs` user which is blacklisted for MapReduce jobs by default.
- Do not use [Hadoop shell commands](#) (such as `cp`, `copyfromlocal`, `put`, `get`) for large copying jobs or you may experience I/O bottlenecks.

DistCp Syntax and Examples

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
```

HFTP Protocol

The HFTP protocol allows you to use FTP resources in an HTTP request. When copying with `distcp` across *different versions of CDH*, use `hftp://` for the source filesystem and `hdfs://` for the destination filesystem, and run `distcp` from the destination cluster. The default port for HFTP is `50070` and the default port for HDFS is `8020`.

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

- `hftp://` is the source protocol.
- `namenode-location` is the CDH 4 (source) NameNode hostname as defined by its configured `fs.default.name`.
- `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`

- `hdfs://` is the destination protocol
- `nameservice-id` or `namenode-location` is the CDH 5 (destination) NameNode hostname as defined by its configured `fs.defaultFS`.
- `basePath` in both examples refers to the directory you want to copy, if one is specifically needed.

Important:

- HFTP is a read-only protocol and can only be used for the source cluster, not the destination.
- HFTP cannot be used when copying with `distcp` from an insecure cluster to a secure cluster.

Using DistCp with Amazon S3

You can copy HDFS files to and from an Amazon S3 instance. You must provision an S3 bucket using Amazon Web Services and obtain the access key and secret key. You can pass these credentials on the `distcp` command line, or you can reference a credential store to "hide" sensitive credentials so that they do not appear in the console output, configuration files, or log files.

Amazon S3 block and native filesystems are supported with the `s3a://` protocol.

Example of an Amazon S3 Block Filesystem URI: `s3a://bucket_name/path/to/file`

S3 credentials can be provided in a configuration file (for example, core-site.xml):

```
<property>
    <name>fs.s3a.access.key</name>
    <value>...</value>
</property>
<property>
    <name>fs.s3a.secret.key</name>
    <value>...</value>
</property>
```

or run on the command line as follows:

```
hadoop distcp -Dfs.s3a.access.key=... -Dfs.s3a.secret.key=... s3a://
```

For example:

```
hadoop distcp -Dfs.s3a.access.key=myAccessKey -Dfs.s3a.secret.key=mySecretKey
hdfs://user/hdfs/mydata s3a://myBucket/mydata_backup
```

> **Important:** Entering secrets on the command line is inherently insecure. These secrets may be accessed in log files and other artifacts. Cloudera recommends that you use a credential provider to store secrets. See Using a Credential Provider to Secure S3 Credentials on page 84.

Using a Credential Provider to Secure S3 Credentials

You can run the `distcp` command without having to enter the access key and secret key on the command line. This prevents these credentials from being exposed in console output, log files, configuration files, and other artifacts. Running the command in this way requires that you provision a credential store to securely store the access key and secret key.

To provision credentials in a credential store:

1. Provision the credentials by running the following commands:

```
hadoop credential create fs.s3a.access.key -value access_key -provider
path_to_credential_store_file
hadoop credential create fs.s3a.secret.key -value secret_key -provider
path_to_credential_store_file
```

   Enter the value for the `-provider` option using the following syntax:

```
provider_type://file/path_to_credential_store_file
```

   provider_type can be one of the following:

   - `jceks` – retrieves credentials from a `JavaKeystoreProvider`. Use when the credential provider is located in HDFS and requires access from multiple hosts.
   - `localjceks` – retrieves credentials from a `LocalJavaKeystoreProvider`. Use when the credential provider is located on the local filesystem and access is only required from this single host. Cloudera recommends `localjceks`.

   For example:

```
hadoop credential create fs.s3a.access.key -value foobar -provider
localjceks://file/home/keystores/aws.jceks
hadoop credential create fs.s3a.secret.key -value barfoo -provider
localjceks://file/home/keystores/aws.jceks
```

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

2. Copy the contents of the `/etc/hadoop/conf` directory to a working directory.
3. Add the following to the `core-site.xml` file in the working directory:

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

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

```
export HADOOP_CONF_DIR=path_to_working_directory
```

After completing these steps, you can run the `distcp` command using the following syntax:

```
hadoop distcp hdfs://source_path s3a://destination_path
```

You can also reference the credential store on the command line, without having to enter it in a copy of the `core-site.xml` file. You also do not need to set a value for `HADOOP_CONF_DIR`. Use the following syntax:

```
hadoop distcp hdfs://source_path s3a://bucket_name/destination_path
-D hadoop.security.credential.provider.path=path_to_credential_store_file
```

There are additional options for the `distcp` command. See [DistCp Guide (Apache Software Foundation)](#).

Examples of DistCP Commands Using the S3 Protocol and Hidden Credentials

Copying files to Amazon S3

```
hadoop distcp hdfs://user/hdfs/mydata s3a://myBucket/mydata_backup
```

Copying files from Amazon S3

```
hadoop distcp s3a://myBucket/mydata_backup hdfs://user/hdfs/mydata
```

Copying files to Amazon S3 using the `-filters` option to exclude specified source files

You specify a file name with the `-filters` option. The referenced file contains regular expressions, one per line, that define file name patterns to exclude from the `distcp` job. The pattern specified in the regular expression should match the fully-qualified path of the intended files, including the scheme (`hdfs`, `webhdfs`, `s3a`, etc.). For example, the following are valid expressions for excluding files:

```
hdfs://x.y.z:8020/a/b/c
webhdfs://x.y.z:50070/a/b/c
s3a://bucket/a/b/c
```

Reference the file containing the filter expressions using `-filters` option. For example:

```
hadoop distcp -filters /user/joe/myFilters hdfs://user/hdfs/mydata
s3a://myBucket/mydata_backup
```

Contents of the sample `myFilters` file:

```
.*foo.*
.*/bar/.*
hdfs://x.y.z:8020/tmp/.*
hdfs://x.y.z:8020/tmp1/file1
```

The regular expressions in the `myFilters` exclude the following files:

- `.*foo.*` – excludes paths that contain the string "`foo`".
- `.*/bar/.*` – excludes paths that include a directory named `bar`.
- `hdfs://x.y.z:8020/tmp/.*` – excludes all files in the `/tmp` directory.

- `hdfs://x.y.z:8020/tmp1/file1` – excludes the file `/tmp1/file1`.

Copying files to Amazon S3 with the `-overwrite` option.

The `-overwrite` option overwrites destination files that already exist.

```
hadoop distcp -overwrite hdfs://user/hdfs/mydata  s3a://user/mydata_backup
```

For more information about the **`-filters`**, **`-overwrite`**, and other options, see [DIstCp Guide: Command Line Options (Apache Software Foundation)](#).

Kerberos Setup Guidelines for Distcp between Secure Clusters

The guidelines mentioned in this section are only applicable for the following sample deployment:

- Let's assume you have two clusters with the realms: `SOURCE` and `DESTINATION`
- You have data that needs to be copied from `SOURCE` to `DESTINATION`
- Trust exists between `SOURCE` and Active Directory, and `DESTINATION` and Active Directory.
- Both `SOURCE` and `DESTINATION` clusters are running CDH 5.3.4 or higher

If your environment matches the one described above, use the following table to configure Kerberos delegation tokens on your cluster so that you can successfully `distcp` across two secure clusters. Based on the direction of the trust between the `SOURCE` and `DESTINATION` clusters, you can use the `mapreduce.job.hdfs-servers.token-renewal.exclude` property to instruct ResourceManagers on either cluster to skip or perform delegation token renewal for NameNode hosts.

<table>
<tr><th colspan="2">Environment Type</th><th>Kerberos Delegation Token Setting</th></tr>
<tr><td rowspan="2"><code>SOURCE</code> trusts <code>DESTINATION</code></td><td>Distcp job runs on the <code>DESTINATION</code> cluster</td><td>You do not need to set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property.</td></tr>
<tr><td>Distcp job runs on the <code>SOURCE</code> cluster</td><td>Set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property to a comma-separated list of the hostnames of the NameNodes of the <code>DESTINATION</code> cluster.</td></tr>
<tr><td rowspan="2"><code>DESTINATION</code> trusts <code>SOURCE</code></td><td>Distcp job runs on the <code>DESTINATION</code> cluster</td><td>Set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property to a comma-separated list of the hostnames of the NameNodes of the <code>SOURCE</code> cluster.</td></tr>
<tr><td>Distcp job runs on the <code>SOURCE</code> cluster</td><td>You do not need to set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property.</td></tr>
<tr><td>Both <code>SOURCE</code> and <code>DESTINATION</code> trust each other</td><td colspan="2">You do not need to set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property.</td></tr>
<tr><td>Neither <code>SOURCE</code> nor <code>DESTINATION</code> trusts the other</td><td colspan="2">If a common realm is usable (such as Active Directory), set the <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property to a comma-separated list of hostnames of the NameNodes of the cluster that is not running the distcp job. For example, if you are running the job on the <code>DESTINATION</code> cluster:
1. <code>kinit</code> on any <code>DESTINATION</code> YARN Gateway host using an AD account that can be used on both <code>SOURCE</code> and <code>DESTINATION</code>.</td></tr>
</table>

<table>
<tr><th>Environment Type</th><th>Kerberos Delegation Token Setting</th></tr>
<tr><td></td><td>
2. Run the distcp job as the hadoop user:
<pre>
$ hadoop distcp
-Ddfs.namenode.kerberos.principal.pattern=*   \
-Dmapreduce.job.hdfs-servers.token-renewal.exclude=SOURCE-nn-host1,SOURCE-nn-host2
   \
hdfs://source-nn-nameservice/source/path   \
/destination/path
</pre>
By default, the YARN ResourceManager renews tokens for applications. The <code>mapreduce.job.hdfs-servers.token-renewal.exclude</code> property instructs ResourceManagers on either cluster to skip delegation token renewal for NameNode hosts.
</td></tr>
</table>

Distcp between Secure Clusters in Distinct Kerberos Realms

This section explains how to copy data between two secure clusters in distinct Kerberos realms.

> **Note:** Both clusters must run a supported JDK version when copying data between Kerberized clusters that are in different realms. For information about supported JDK versions, see CDH 5 and Cloudera Manager 5 Requirements and Supported 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_enctypes = 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
```

Configure HDFS RPC Protection and Acceptable Kerberos Principal Patterns

Set the `hadoop.rpc.protection` property to `authentication` in both clusters. You can modify this property either in `hdfs-site.xml`, or using Cloudera Manager as follows:

1. Open the Cloudera Manager Admin Console.
2. Go to the HDFS service.
3. Click the **Configuration** tab.
4. Select **Scope** > **HDFS-1 (Service-Wide)**
5. Select **Category** > **Security**.
6. Locate the **Hadoop RPC Protection** property and select `authentication`.
7. Click **Save Changes** to commit the changes.

The following steps are not required if the two realms are already set up to trust each other, or have the same principal pattern. However, this isn't usually the case.

Set the `dfs.namenode.kerberos.principal.pattern` property to * to allow distcp irrespective of the principal patterns of the source and destination clusters. You can modify this property either in `hdfs-site.xml` on both clusters, or using Cloudera Manager as follows:

1. Open the Cloudera Manager Admin Console.
2. Go to the HDFS service.
3. Click the **Configuration** tab.
4. Select **Scope** > **Gateway**
5. Select **Category** > **Advanced**.
6. Edit the **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** property to add:

```
<property>
  <name>dfs.namenode.kerberos.principal.pattern</name>
  <value>*</value>
</property>
```

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

(If TLS/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
```

Enabling Fallback Configuration

To enable the fallback configuration, for copying between secure and insecure clusters, add the following to the HDFS configuration file, `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>
```

Protocol Support for Distcp

The following table lists the protocols supported with the `distcp` command on different versions of CDH. "Secure" means that the cluster is configured to use Kerberos.

> **Note:** Copying between a secure cluster and an insecure cluster is only supported with CDH 5.1.3 and higher (CDH 5.1.3+) in accordance with HDFS-6776.

Source	Destination	Where to Issue distcp Command	Source Protocol	Source Config	Destination Protocol	Destination Config	Fallback Config Required
CDH 4	CDH 4	Destination	hftp	Secure	hdfs or webhdfs	Secure	
CDH 4	CDH 4	Source or Destination	hdfs or webhdfs	Secure	hdfs or webhdfs	Secure	
CDH 4	CDH 4	Source or Destination	hdfs or webhdfs	Insecure	hdfs or webhdfs	Insecure	
CDH 4	CDH 4	Destination	hftp	Insecure	hdfs or webhdfs	Insecure	
CDH 4	CDH 5	Destination	webhdfs or hftp	Secure	webhdfs or hdfs	Secure	
CDH 4	CDH 5.1.3+	Destination	webhdfs	Insecure	webhdfs	Secure	Yes
CDH 4	CDH 5	Destination	webhdfs or hftp	Insecure	webhdfs or hdfs	Insecure	
CDH 4	CDH 5	Source	hdfs or webhdfs	Insecure	webhdfs	Insecure	
CDH 5	CDH 4	Source or Destination	webhdfs	Secure	webhdfs	Secure	
CDH 5	CDH 4	Source	hdfs	Secure	webhdfs	Secure	
CDH 5.1.3+	CDH 4	Source	hdfs or webhdfs	Secure	webhdfs	Insecure	Yes
CDH 5	CDH 4	Source or Destination	webhdfs	Insecure	webhdfs	Insecure	
CDH 5	CDH 4	Destination	webhdfs	Insecure	hdfs	Insecure	
CDH 5	CDH 4	Source	hdfs	Insecure	webhdfs	Insecure	
CDH 5	CDH 4	Destination	hftp	Insecure	hdfs or webhdfs	Insecure	
CDH 5	CDH 5	Source or Destination	hdfs or webhdfs	Secure	hdfs or webhdfs	Secure	
CDH 5	CDH 5	Destination	hftp	Secure	hdfs or webhdfs	Secure	

Source	Destination	Where to Issue distcp Command	Source Protocol	Source Config	Destination Protocol	Destination Config	Fallback Config Required
CDH 5.1.3+	CDH 5	Source	hdfs or webhdfs	Secure	hdfs or webhdfs	Insecure	Yes
CDH 5	CDH 5.1.3+	Destination	hdfs or webhdfs	Insecure	hdfs or webhdfs	Secure	Yes
CDH 5	CDH 5	Source or Destination	hdfs or webhdfs	Insecure	hdfs or webhdfs	Insecure	
CDH 5	CDH 5	Destination	hftp	Insecure	hdfs or webhdfs	Insecure	

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. Note that when doing this, the `distcp` commands should be run from the secure cluster. by doing the following:

1. On the secure cluster, set `ipc.client.fallback-to-simple-auth-allowed` to true in core-site.xml:

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

2. On the insecure cluster, add the secured cluster's realm name to the insecure cluster's configuration:
   a. In the Cloudera Manager Admin Console for the insecure cluster, navigate to **Clusters** > **<HDFS cluster>**.
   b. On the **Configuration** tab, search for **Trusted Kerberos Realms** and add the secured cluster's realm name.

      Note that his does not require Kerberos to be enabled but is a necessary step to allow the simple auth fallback to happen in the `hdfs://` protocol.
   c. Save the change.
3. 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.

Decommissioning DataNodes Using the Command Line

Decommissioning a DataNode excludes it from a cluster after its data is replicated to active nodes. To decommision a DataNode:

1. Create a file named `dfs.exclude` in the *HADOOP_CONF_DIR* (default is `/etc/hadoop/conf`).
2. Add the name of each DataNode host to be decommissioned on individual lines.
3. Stop the TaskTracker on the DataNode to be decommissioned.
4. Add the following property to `hdfs-site.xml` on the NameNode host.

```
<property>
  <name>dfs.hosts.exclude</name>
```

```
  <value>/etc/hadoop/conf/dfs.exclude</value>
<property>
```

When a DataNode is marked for decommission, all of the blocks on that DataNode are marked as *under replicated*. In the NameNode UI under Decommissioning DataNodes you can see a total number of under replicated blocks, which will reduce over time, indicating decommissioning progress.

Cloudera recommends that you decommission no more than two DataNodes at one time.

Stopping the Decommissioning Process

To stop the decommissioning process for a DataNode using the command line:

1. Remove the DataNode name from `/etc/hadoop/conf/dfs.exclude`.
2. Run the command `$ hdfs dfsadmin -refreshNodes`.

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 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 box 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 box.

   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 Using Cloudera Manager on page 13.
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 Using Cloudera Manager on page 13.

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 247 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**.

Using Optimal Message Sizes with Flume

For best performance, Cloudera recommends you configure your applications to send messages smaller than 2 MiB in size through Flume.

Backing Up Flume Channel Data Directories

A best practice is to periodically back up your Flume data directories. The `dataDir` and `checkpointDir` are located in your Flume home directory: its default location is `/var/lib/flume-ng`. You can verify the home directory location in Cloudera Manager by going to the **Flume** > **Configuration** tab and searching for the field **Flume Home Directory**.

To back up Flume data directories:

1. In Cloudera Manager, go to the Flume service.
2. Stop Flume to ensure that no changes are written to the data or checkpoint directories during the backup.
3. Perform a file-level backup of the `dataDir` and `checkpointDir`.
4. Restart Flume.

For more information on starting and stopping services, see Starting, Stopping, and Restarting Services.

Configuring Flume Security with Kafka

Cloudera Manager does not provide configuration options for Flume to work with Kafka sources and channels over TLS. When Kafka is configured with TLS, additional manual configuration is required to communicate with Flume.

This topic describes how to configure Flume to communicate with Kafka TLS.

Set Up Cloudera Manager to Generate `flume.keytab`

If you have already set up Kerberos because of a kerberized HDFS or Solr dependency on Flume, then this step is already done. Verify your Kerberos keytab files.

If you do not have a Kerberos dependency or the key tab files haven't been generated, then extend your Flume configuration. If the agent requires Kerberos credentials, then the configuration file must have a line that has the following attributes:

- Begins with the agent's name.
- Contains the `$KERBEROS_PRINCIPAL` string.
- Is syntactically correct.

For example:

```
tier1.sources.source1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

The property name *generateKeyTabFor* is an arbitrary name that is not used by Flume. There is no dependency on HDFS or any other service.

Verify the flume.keytab

After you configure and restart the agent, the key tab file is generated at the following directory location:

```
/var/run/cloudera-scm-agent/process/<latest_id>-flume-AGENT/flume.keytab
```

The file must not be empty on any host that runs a kerberized Flume agent.

Principal managaement is handled by Cloudera Manager for Flume, just as with other services. For example, principals are listed on the **Administration** > **Security** > **Kerberos Credentials** page in Cloudera Manager.

Create jaas.conf

Create a `flafka_jaas.conf` file on each host that runs a Flume agent. The configuration information is used to communicate with Kafka and also provide normal Flume Kerberos support. The `flafka_jaas.conf` file contains two entries for the Flume principal: *Client* and *KafkaClient*. Note that the *principal* property is host specific. Unix user `flume` must have read permission for this file.

```
/opt/cloudera/security/flafka_jaas.conf:
    Client {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    keyTab="flume.keytab"
    principal="flume/cornhost-1.gce.acmecorn.com@GCE.ACMECORN.COM";
    };

    KafkaClient {
    com.sun.security.auth.module.Krb5LoginModule required
    useKeyTab=true
    storeKey=true
    serviceName="kafka"
    keyTab="flume.keytab"
    principal="flume/cornhost-1.gce.acmecorn.com@GCE.ACMECORN.COM";
    };
```

Set Up jaas for Flume in Cloudera Manager

In Flume service configuration, the **Java Configuration Options for FlumeAgent** requires the following:

```
-Djava.security.auth.login.config=/opt/cloudera/security/flafka_jaas.conf
```

Do not use **Flume Service Environment Advanced Configuration Snippet (Safety Valve)** to set this property using *FLUME_AGENT_JAVA_OPTS*, as it will override existing Java command line options.

Update Flume Service Configuration Kerberos Authentication and TLS Encryption

Add the relevant properties to the Flume source or Flume channel, depending on the broker.protocol type (in this case, SASL_SSL).

The default secure port of Kafka brokers is 9093 instead of 9092. Update the kafka.bootstrap.servers as well.

Kafka Source

Update the Flume Kafka source entries to include the following security configuration.

```
tier1.sources.source1.kafka.consumer.security.protocol = SASL_SSL
tier1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
tier1.sources.source1.kafka.consumer.ssl.truststore.location=/opt/cloudera/security/jks/truststore.jks
tier1.sources.source1.kafka.consumer.ssl.truststore.password=cloudera
tier1.sources.source1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Kafka Channel

Update the Flume Kafka channel entries to include the following security configuration. Replace the truststore location and password with the correct path for the cluster. Both producer and consumer configurations are required.

```
tier1.channels.channel1.kafka.consumer.security.protocol = SASL_SSL
tier1.channels.channel1.kafka.consumer.sasl.kerberos.service.name = kafka
tier1.channels.channel1.kafka.consumer.ssl.truststore.location=/opt/cloudera/security/jks/truststore.jks

tier1.channels.channel1.kafka.consumer.ssl.truststore.password=cloudera

tier1.channels.channel1.kafka.producer.security.protocol = SASL_SSL
tier1.channels.channel1.kafka.producer.sasl.kerberos.service.name = kafka
tier1.channels.channel1.kafka.producer.ssl.truststore.location=/opt/cloudera/security/jks/truststore.jks

tier1.channels.channel1.kafka.producer.ssl.truststore.password=cloudera
tier1.channels.channel1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Kafka Sink

Update the Flume Kafka sink entries to include the following security configuration.

```
tier1.sinks.sink1.kafka.producer.security.protocol = SASL_SSL
tier1.sinks.sink1.kafka.producer.sasl.kerberos.service.name = kafka
tier1.sinks.sink1.kafka.producer.ssl.truststore.location=/opt/cloudera/security/jks/truststore.jks
tier1.sinks.sink1.kafka.producer.ssl.truststore.password=cloudera
tier1.sinks.sink1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Update Flume Service Configuration: Kerberos Authentication with No Encryption

Add the relevant properties to the Flume source or Flume channel, depending on the *broker.protocol type* (in this case, *SASL_PLAINTEXT*).

The default secure port of Kafka brokers is 9093 rather than 9092: update *kafka.bootstrap.servers* as well.

Kafka Source

Update the Flume Kafka source entries to include the following security configuration. Replace the truststore location and password with the correct path for the cluster.

```
tier1.sources.source1.kafka.consumer.security.protocol = SASL_PLAINTEXT
tier1.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
tier1.sources.source1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Kafka Channel

Update the Flume Kafka channel entries to include the following security configuration. Replace the truststore location and password with the correct path for the cluster. Both producer and consumer configurations are required.

```
tier1.channels.channel1.kafka.consumer.security.protocol = SASL_PLAINTEXT
tier1.channels.channel1.kafka.consumer.sasl.kerberos.service.name = kafka

tier1.channels.channel1.kafka.producer.security.protocol = SASL_PLAINTEXT
tier1.channels.channel1.kafka.producer.ssl.truststore.password=cloudera
tier1.channels.channel1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Kafka Sink

Update the Flume Kafka sink entries to include the following security configuration. Replace the truststore location and password with the correct path for the cluster.

```
tier1.sinks.sink1.kafka.producer.security.protocol = SASL_PLAINTEXT
tier1.sinks.sink1.kafka.producer.sasl.kerberos.service.name = kafka
tier1.sinks.sink1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

Example

The code sample below is a complete working example Flume configuration with two tiers. Tier1 reads an input log and puts the new Events to the *sectest* topic using a Kafka Sink (the tailed file has to exist before agent starts). Tier2 listens to the *sectest* topic by a Kafka Source and logs every event.

```
#########################################################
# Tier1
#########################################################
tier1.sources  = source1
tier1.channels = channel1
tier1.sinks    = sink1

tier1.channels.channel1.type   = memory
tier1.channels.channel1.capacity = 1000
tier1.channels.channel1.transactionCapacity = 100

tier1.sinks.sink1.channel      = channel1
tier1.sources.source1.channels = channel1

tier1.sources.source1.type  = exec
tier1.sources.source1.command  = tail -F /tmp/input/input.log
tier1.sinks.sink1.type = org.apache.flume.sink.kafka.KafkaSink
tier1.sinks.sink1.kafka.topic = sectest

tier1.sinks.sink1.kafka.bootstrap.servers =
acmecp-ssl-1.gce.cloudera.com:9093,acmecp-ssl-2.gce.cloudera.com:9093

###
# Security related setup
###
tier1.sinks.sink1.kafka.producer.security.protocol = SASL_SSL
tier1.sinks.sink1.kafka.producer.sasl.kerberos.service.name = kafka
tier1.sinks.sink1.kafka.producer.ssl.truststore.location=/etc/cdep-ssl-conf/CA_STANDARD/truststore.jks
tier1.sinks.sink1.kafka.producer.ssl.truststore.password=cloudera
tier1.sinks.sink1.generateKeytabFor = $KERBEROS_PRINCIPAL

#########################################################
# Tier2
#########################################################
tier2.sources  = source1
tier2.channels = channel1
tier2.sinks    = sink1

tier2.channels.channel1.type   = memory
tier2.channels.channel1.capacity = 1000
tier2.channels.channel1.transactionCapacity = 100

tier2.sinks.sink1.channel      = channel1
tier2.sources.source1.channels = channel1

tier2.sources.source1.type  = org.apache.flume.source.kafka.KafkaSource
tier2.sources.source1.kafka.bootstrap.servers =
acmecp-ssl-1.gce.cloudera.com:9093,acmecp-ssl-2.gce.cloudera.com:9093
tier2.sources.source1.kafka.topics = sectest
tier2.sources.source1.kafka.offsets.storage = kafka
tier2.sources.source1.kafka.consumer.group.id = flume
tier2.sources.source1.kafka.consumer.auto.offset.reset = earliest
tier2.sinks.sink1.type = logger

###
# Security related setup
```

```
###
tier2.sources.source1.kafka.consumer.security.protocol = SASL_SSL
tier2.sources.source1.kafka.consumer.sasl.kerberos.service.name = kafka
tier2.sources.source1.kafka.consumer.ssl.truststore.location=/etc/cdep-ssl-conf/CA_STANDARD/truststore.jks
tier2.sources.source1.kafka.consumer.ssl.truststore.password=cloudera
tier2.sources.source1.generateKeytabFor = $KERBEROS_PRINCIPAL
```

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.

Disabling Loading of Coprocessors

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

In CDH 5.7 and higher, you can disable loading of system (HBase-wide) or user (table-wide) coprocessors. Cloudera recommends against disabling loading of system coprocessors, because HBase security functionality is implemented using system coprocessors. However, disabling loading of user coprocessors may be appropriate.

1. Select the HBase service.
2. Click the **Configuration** tab.
3. Search for **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml**.
4. To disable loading of all coprocessors, add a new property with the name `hbase.coprocessor.enabled` and set its value to `false`. **Cloudera does not recommend this setting.**
5. To disable loading of user coprocessors, add a new property with the name `hbase.coprocessor.user.enabled` and set its value to `false`.
6. 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.

Managing HBase Security

This topic pulls together content also found elsewhere which relates to configuring and using HBase in a secure environment. For the most part, securing an HBase cluster is a one-way operation, and moving from a secure to an unsecure configuration should not be attempted without contacting Cloudera support for guidance.

HBase Authentication

To configure HBase security, complete the following tasks:

1. **Configure HBase Authentication**: You must establish a mechanism for HBase servers and clients to securely identify themselves with HDFS, ZooKeeper, and each other. This ensures that hosts are who they claim to be.

   > **Note:**
   >
   > - To enable HBase to work with Kerberos security, you must perform the installation and configuration steps in Configuring Hadoop Security in CDH 5 and ZooKeeper Security Configuration.
   > - Although an HBase Thrift server can connect to a secured Hadoop cluster, access is not secured from clients to the HBase Thrift server. To encrypt communication between clients and the HBase Thrift Server, see Configuring TLS/SSL for HBase Thrift Server.

   The following sections describe how to use Apache HBase and CDH 5 with Kerberos security:

   - Configuring Kerberos Authentication for HBase
   - Configuring Secure HBase Replication
   - Configuring the HBase Client TGT Renewal Period

2. **Configure HBase Authorization**: You must establish rules for the resources that clients are allowed to access. For more information, see Configuring HBase Authorization.

Using the Hue HBase App

Hue includes an HBase App that allows you to interact with HBase through a Thrift proxy server. Because Hue sits between the Thrift server and the client, the Thrift server assumes that all HBase operations come from the hue user and not the client. To ensure that users in Hue are only allowed to perform HBase operations assigned to their own credentials, and not those of the hue user, you must enable HBase impersonation. For more information about the how to enable doAs Impersonation for the HBase Browser Application, see Enabling the HBase Browser Application with doAs Impersonation on page 224.

Configuring HBase Authorization

After configuring HBase authentication (as detailed in HBase Configuration), you must define rules on resources that is allowed to access. HBase rules can be defined for individual tables, columns, and cells within a table. Cell-level authorization is fully supported since CDH 5.2.

> **Important:** In a cluster managed by Cloudera Manager, HBase authorization is disabled by default. You have to enable HBase authorization (as detailed in Enable HBase Authorization) to use any kind of authorization method.

Understanding HBase Access Levels

HBase access levels are granted independently of each other and allow for different types of operations at a given scope.

- **Read (R)** - can read data at the given scope
- **Write (W)** - can write data at the given scope
- **Execute (X)** - can execute coprocessor endpoints at the given scope
- **Create (C)** - can create tables or drop tables (even those they did not create) at the given scope

- **Admin (A)** - can perform cluster operations such as balancing the cluster or assigning regions at the given scope

The possible scopes are:

- **Superuser** - superusers can perform any operation available in HBase, to any resource. The user who runs HBase on your cluster is a superuser, as are any principals assigned to the configuration property `hbase.superuser` in `hbase-site.xml` on the HMaster.
- **Global** - permissions granted at `global` scope allow the admin to operate on all tables of the cluster.
- **Namespace** - permissions granted at `namespace` scope apply to all tables within a given namespace.
- **Table** - permissions granted at `table` scope apply to data or metadata within a given table.
- **ColumnFamily** - permissions granted at `ColumnFamily` scope apply to cells within that ColumnFamily.
- **Cell** - permissions granted at Cell scope apply to that exact cell coordinate. This allows for policy evolution along with data. To change an ACL on a specific cell, write an updated cell with new ACL to the precise coordinates of the original. If you have a multi-versioned schema and want to update ACLs on all visible versions, you'll need to write new cells for all visible versions. The application has complete control over policy evolution. The exception is `append` and `increment` processing. `Appends` and `increments` can carry an ACL in the operation. If one is included in the operation, then it will be applied to the result of the `append` or `increment`. Otherwise, the ACL of the existing cell being appended to or incremented is preserved.

The combination of access levels and scopes creates a matrix of possible access levels that can be granted to a user. In a production environment, it is useful to think of access levels in terms of what is needed to do a specific job. The following list describes appropriate access levels for some common types of HBase users. It is important not to grant more access than is required for a given user to perform their required tasks.

- **Superusers** - In a production system, only the HBase user should have superuser access. In a development environment, an administrator might need superuser access to quickly control and manage the cluster. However, this type of administrator should usually be a `Global Admin` rather than a superuser.
- **Global Admins** - A `global admin` can perform tasks and access every table in HBase. In a typical production environment, an admin should not have `Read` or `Write` permissions to data within tables.
  - A global admin with `Admin` permissions can perform cluster-wide operations on the cluster, such as balancing, assigning or unassigning regions, or calling an explicit major compaction. This is an operations role.
  - A global admin with `Create` permissions can create or drop any table within HBase. This is more of a DBA-type role.

  In a production environment, it is likely that different users will have only one of `Admin` and `Create` permissions.

  > **Warning:**
  >
  > In the current implementation, a `Global Admin` with `Admin` permission can grant himself `Read` and `Write` permissions on a table and gain access to that table's data. For this reason, only grant `Global Admin` permissions to trusted user who actually need them.
  >
  > Also be aware that a `Global Admin` with `Create` permission can perform a `Put` operation on the ACL table, simulating a `grant` or `revoke` and circumventing the authorization check for `Global Admin` permissions. This issue (but not the first one) is fixed in CDH 5.3 and higher, as well as CDH 5.2.1. It is not fixed in CDH 4.x or CDH 5.1.x.
  >
  > Due to these issues, be cautious with granting Global Admin privileges.

- **Namespace Admin** - a namespace admin with Create permissions can create or drop tables within that namespace, and take and restore snapshots. A namespace admin with Admin permissions can perform operations such as splits or major compactions on tables within that namespace. Prior to CDH 5.4, only global admins could create namespaces. In CDH 5.4, any user with Namespace Create privileges can create namespaces.
- **Table Admins** - A table admin can perform administrative operations only on that table. A table admin with `Create` permissions can create snapshots from that table or restore that table from a snapshot. A table admin with `Admin` permissions can perform operations such as splits or major compactions on that table.

- **Users** - Users can read or write data, or both. Users can also execute coprocessor endpoints, if given `Executable` permissions.

> **Important:**
>
> If you are using Kerberos principal names when setting ACLs for users, Hadoop uses only the first part (short) of the Kerberos principal when converting it to the username. Hence, for the principal `ann/fully.qualified.domain.name@YOUR-REALM.COM`, HBase ACLs should only be set for user `ann`.

Table 5: Real-World Example of Access Levels

This table shows some typical job descriptions at a hypothetical company and the permissions they might require to get their jobs done using HBase.

Job Title	Scope	Permissions	Description
Senior Administrator	Global	Admin, Create	Manages the cluster and gives access to Junior Administrators.
Junior Administrator	Global	Create	Creates tables and gives access to Table Administrators.
Table Administrator	Table	Admin	Maintains a table from an operations point of view.
Data Analyst	Table	Read	Creates reports from HBase data.
Web Application	Table	Read, Write	Puts data into HBase and uses HBase data to perform operations.

Further Reading

- Access Control Matrix
- Security - Apache HBase Reference Guide

Enable HBase Authorization

HBase authorization is built on top of the Coprocessors framework, specifically `AccessController Coprocessor`.

> **Note:** Once the Access Controller coprocessor is enabled, any user who uses the HBase shell will be subject to access control. Access control will also be in effect for native (Java API) client access to HBase.

Enable HBase Authorization Using Cloudera Manager

1. Go to **Clusters** and select the HBase cluster.
2. Select **Configuration**.
3. Search for **HBase Secure Authorization** and select it.
4. Search for **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml** and enter the following into it to enable `hbase.security.exec.permission.checks`. Without this option, all users will continue to

have access to execute endpoint coprocessors. This option is not enabled when you enable HBase Secure Authorization for backward compatibility.

```
<property>
  <name>hbase.security.exec.permission.checks</name>
  <value>true</value>
</property>
```

5. Optionally, search for and configure **HBase Coprocessor Master Classes** and **HBase Coprocessor Region Classes**.

Enable HBase Authorization Using the Command Line

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

To enable HBase authorization, add the following properties to the `hbase-site.xml` file *on every HBase server host (Master or RegionServer)*:

```
<property>
     <name>hbase.security.authorization</name>
     <value>true</value>
</property>
<property>
  <name>hbase.security.exec.permission.checks</name>
  <value>true</value>
</property>
<property>
     <name>hbase.coprocessor.master.classes</name>
     <value>org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
<property>
     <name>hbase.coprocessor.region.classes</name>

<value>org.apache.hadoop.hbase.security.token.TokenProvider,org.apache.hadoop.hbase.security.access.AccessController</value>
</property>
```

Configure Access Control Lists for Authorization

Now that HBase has the security coprocessor enabled, you can set ACLs using the HBase shell. Start the HBase shell as usual.

> **Important:**
>
> The host running the shell must be configured with a keytab file as described in Configuring Kerberos Authentication for HBase.

The commands that control ACLs take the following form. Group names are prefixed with the @ symbol.

```
hbase> grant <user> <permissions> [ @<namespace> [ <table>[ <column family>[ <column
qualifier> ] ] ] ]   # grants permissions

hbase> revoke <user> [ @<namespace> [ <table> [ <column family> [ <column qualifier> ]
 ] ]  # revokes permissions

hbase> user_permission <table>
                    # displays existing permissions
```

In the above commands, fields encased in <> are variables, and fields in [] are optional. The `permissions` variable must consist of zero or more character from the set "RWCA".

- `R` denotes read permissions, which is required to perform `Get`, `Scan`, or `Exists` calls in a given scope.
- `W` denotes write permissions, which is required to perform `Put`, `Delete`, `LockRow`, `UnlockRow`, `IncrementColumnValue`, `CheckAndDelete`, `CheckAndPut`, `Flush`, or `Compact` in a given scope.
- `X` denotes execute permissions, which is required to execute coprocessor endpoints.
- `C` denotes create permissions, which is required to perform `Create`, `Alter`, or `Drop` in a given scope.
- `A` denotes admin permissions, which is required to perform `Enable`, `Disable`, `Snapshot`, `Restore`, `Clone`, `Split`, `MajorCompact`, `Grant`, `Revoke`, and `Shutdown` in a given scope.

Access Control List Example Commands

```
grant 'user1', 'RWC'
grant 'user2', 'RW', 'tableA'
grant 'user3', 'C', '@my_namespace'
```

Be sure to review the information in Understanding HBase Access Levels to understand the implications of the different access levels.

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**.

Other HBase Security Topics

- Using BulkLoad On A Secure Cluster on page 143
- Configuring Secure HBase Replication

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 97.
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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

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

1. 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 HMasters, followed by the main HMaster, by using the `hbase-master stop` command.

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

Accessing HBase by using the HBase Shell

After you have started HBase, you can access the database in an interactive way by using the HBase Shell, which is a command interpreter for HBase which is written in Ruby. Always run HBase administrative commands such as the HBase Shell, `hbck`, or bulk-load commands as the HBase user (typically `hbase`).

```
$ hbase shell
```

HBase Shell Overview

- To get help and to see all available commands, use the `help` command.
- To get help on a specific command, use `help "command"`. For example:

```
hbase> help "create"
```

- To remove an attribute from a table or column family or reset it to its default value, set its value to `nil`. For example, use the following command to remove the `KEEP_DELETED_CELLS` attribute from the `f1` column of the `users` table:

```
hbase> alter 'users', { NAME => 'f1', KEEP_DELETED_CELLS => nil }
```

- To exit the HBase Shell, type `quit`.

Setting Virtual Machine Options for HBase Shell

HBase in CDH 5.2 and higher allows you to set variables for the virtual machine running HBase Shell, by using the HBASE_SHELL_OPTS environment variable. This example sets several options in the virtual machine.

```
$ HBASE_SHELL_OPTS="-verbose:gc -XX:+PrintGCApplicationStoppedTime -XX:+PrintGCDateStamps

      -XX:+PrintGCDetails -Xloggc:$HBASE_HOME/logs/gc-hbase.log" ./bin/hbase shell
```

Scripting with HBase Shell

CDH 5.2 and higher include non-interactive mode. This mode allows you to use HBase Shell in scripts, and allow the script to access the exit status of the HBase Shell commands. To invoke non-interactive mode, use the `-n` or `--non-interactive` switch. This small example script shows how to use HBase Shell in a Bash script.

```
#!/bin/bash
echo 'list' | hbase shell -n
status=$?
if [$status -ne 0]; then
  echo "The command may have failed."
fi
```

Successful HBase Shell commands return an exit status of 0. However, an exit status other than 0 does not necessarily indicate a failure, but should be interpreted as unknown. For example, a command may succeed, but while waiting for the response, the client may lose connectivity. In that case, the client has no way to know the outcome of the command. In the case of a non-zero exit status, your script should check to be sure the command actually failed before taking further action.

CDH 5.7 and higher include the `get_splits` command, which returns the split points for a given table:

```
hbase> get_splits 't2'
Total number of splits = 5

=> ["", "10", "20", "30", "40"]
```

You can also write Ruby scripts for use with HBase Shell. Example Ruby scripts are included in the `hbase-examples/src/main/ruby/` directory.

Using HBase Command-Line Utilities

Besides the [HBase Shell](), HBase includes several other command-line utilities, which are available in the `hbase/bin/` directory of each HBase host. This topic provides basic usage instructions for the most commonly used utilities.

`PerformanceEvaluation`

The `PerformanceEvaluation` utility allows you to run several preconfigured tests on your cluster and reports its performance. To run the `PerformanceEvaluation` tool in CDH 5.1 and higher, use the `bin/hbase pe` command. In CDH 5.0 and lower, use the command `bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation`.

For usage instructions, run the command with no arguments. The following output shows the usage instructions for the `PerformanceEvaluation` tool in CDH 5.7. Options and commands available depend on the CDH version.

```
$ hbase pe

Usage: java org.apache.hadoop.hbase.PerformanceEvaluation \
  <OPTIONS> [-D<property=value>]* <command> <nclients>

Options:
 nomapred        Run multiple clients using threads (rather than use mapreduce)
 rows            Rows each client runs. Default: One million
 size            Total size in GiB. Mutually exclusive with --rows. Default: 1.0.
 sampleRate      Execute test on a sample of total rows. Only supported by randomRead.
                 Default: 1.0
 traceRate       Enable HTrace spans. Initiate tracing every N rows. Default: 0
 table           Alternate table name. Default: 'TestTable'
 multiGet        If >0, when doing RandomRead, perform multiple gets instead of single
                 gets.
                 Default: 0
 compress        Compression type to use (GZ, LZO, ...). Default: 'NONE'
 flushCommits    Used to determine if the test should flush the table. Default: false
 writeToWAL      Set writeToWAL on puts. Default: True
 autoFlush       Set autoFlush on htable. Default: False
 oneCon          all the threads share the same connection. Default: False
 presplit        Create presplit table. Recommended for accurate perf analysis (see
                 guide). Default: disabled
 inmemory        Tries to keep the HFiles of the CF inmemory as far as possible. Not
                 guaranteed that reads are always served from memory.  Default: false
 usetags         Writes tags along with KVs. Use with HFile V3. Default: false
 numoftags       Specify the no of tags that would be needed. This works only if usetags
                 is true.
 filterAll       Helps to filter out all the rows on the server side there by not returning
                 anything back to the client.  Helps to check the server side performance.
                 Uses FilterAllFilter internally.
 latency         Set to report operation latencies. Default: False
 bloomFilter     Bloom filter type, one of [NONE, ROW, ROWCOL]
 valueSize       Pass value size to use: Default: 1024
 valueRandom     Set if we should vary value size between 0 and 'valueSize'; set on read
                 for stats on size: Default: Not set.
 valueZipf       Set if we should vary value size between 0 and 'valueSize' in zipf form:
                 Default: Not set.
 period          Report every 'period' rows: Default: opts.perClientRunRows / 10
 multiGet        Batch gets together into groups of N. Only supported by randomRead.
                 Default: disabled
 addColumns      Adds columns to scans/gets explicitly. Default: true
 replicas        Enable region replica testing. Defaults: 1.
 splitPolicy     Specify a custom RegionSplitPolicy for the table.
 randomSleep     Do a random sleep before each get between 0 and entered value. Defaults: 0
 columns         Columns to write per row. Default: 1
 caching         Scan caching to use. Default: 30

 Note: -D properties will be applied to the conf used.
  For example:
   -Dmapreduce.output.fileoutputformat.compress=true
   -Dmapreduce.task.timeout=60000
```

```
Command:
 append          Append on each row; clients overlap on keyspace so some concurrent
                 operations
 checkAndDelete  CheckAndDelete on each row; clients overlap on keyspace so some concurrent
                 operations
 checkAndMutate  CheckAndMutate on each row; clients overlap on keyspace so some concurrent
                 operations
 checkAndPut     CheckAndPut on each row; clients overlap on keyspace so some concurrent
                 operations
 filterScan      Run scan test using a filter to find a specific row based on it's value
                 (make sure to use --rows=20)
 increment       Increment on each row; clients overlap on keyspace so some concurrent
                 operations
 randomRead      Run random read test
 randomSeekScan  Run random seek and scan 100 test
 randomWrite     Run random write test
 scan            Run scan test (read every row)
 scanRange10     Run random seek scan with both start and stop row (max 10 rows)
 scanRange100    Run random seek scan with both start and stop row (max 100 rows)
 scanRange1000   Run random seek scan with both start and stop row (max 1000 rows)
 scanRange10000  Run random seek scan with both start and stop row (max 10000 rows)
 sequentialRead  Run sequential read test
 sequentialWrite Run sequential write test

Args:
 nclients        Integer. Required. Total number of clients (and HRegionServers)
                 running: 1 <= value <= 500
Examples:
 To run a single client doing the default 1M sequentialWrites:
 $ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation sequentialWrite 1
 To run 10 clients doing increments over ten rows:
 $ bin/hbase org.apache.hadoop.hbase.PerformanceEvaluation --rows=10 --nomapred increment 10
```

LoadTestTool

The LoadTestTool utility load-tests your cluster by performing writes, updates, or reads on it. To run the LoadTestTool in CDH 5.1 and higher, use the bin/hbase ltt command . In CDH 5.0 and lower, use the command bin/hbase org.apache.hadoop.hbase.util.LoadTestTool. To print general usage information, use the -h option. Options and commands available depend on the CDH version.

```
$ bin/hbase ltt -h

Options:
 -batchupdate                    Whether to use batch as opposed to separate updates for every
column
                                 in a row
 -bloom <arg>                    Bloom filter type, one of [NONE, ROW, ROWCOL]
 -compression <arg>              Compression type, one of [LZO, GZ, NONE, SNAPPY, LZ4]
 -data_block_encoding <arg>      Encoding algorithm (e.g. prefix compression) to use for data
blocks
                                 in the test column family, one of
                                 [NONE, PREFIX, DIFF, FAST_DIFF, PREFIX_TREE].
 -deferredlogflush               Enable deferred log flush.
 -encryption <arg>               Enables transparent encryption on the test table, one of [AES]
 -families <arg>                 The name of the column families to use separated by comma
 -generator <arg>                The class which generates load for the tool. Any args for this
class
                                 can be passed as colon separated after class name
 -h,--help                       Show usage
 -in_memory                      Tries to keep the HFiles of the CF inmemory as far as possible.
 Not
                                 guaranteed that reads are always served from inmemory
 -init_only                      Initialize the test table only, don't do any loading
 -key_window <arg>            The 'key window' to maintain between reads and writes for concurrent

                                 write/read workload. The default is 0.
 -max_read_errors <arg>          The maximum number of read errors to tolerate before terminating
 all
                                 reader threads. The default is 10.
 -mob_threshold <arg>            Desired cell size to exceed in bytes that will use the MOB write
 path
 -multiget_batchsize <arg>       Whether to use multi-gets as opposed to separate gets for every
                                 column in a row
 -multiput                       Whether to use multi-puts as opposed to separate puts for every
                                 column in a row
 -num_keys <arg>                 The number of keys to read/write
 -num_regions_per_server <arg>   Desired number of regions per region server. Defaults to 5.
 -num_tables <arg>               A positive integer number. When a number n is speicfied, load
```

```
test tool
                                 will load n table parallely. -tn parameter value becomes table
name prefix.
                                 Each table name is in format <tn>_1...<tn>_n
 -read <arg>                     <verify_percent>[:<#threads=20>]
 -reader <arg>                   The class for executing the read requests
 -region_replica_id <arg>        Region replica id to do the reads from
 -region_replication <arg>       Desired number of replicas per region
 -regions_per_server <arg>       A positive integer number. When a number n is specified, load
test tool
                                 will create the test table with n regions per server
 -skip_init                      Skip the initialization; assume test table already exists
 -start_key <arg>                The first key to read/write (a 0-based index). The default value
 is 0.
 -tn <arg>                       The name of the table to read or write
 -update <arg>                   <update_percent>[:<#threads=20>][:<#whether to ignore nonce
collisions=0>]
 -updater <arg>                  The class for executing the update requests
 -write <arg>                    <avg_cols_per_key>:<avg_data_size>[:<#threads=20>]
 -writer <arg>                   The class for executing the write requests
 -zk <arg>                       ZK quorum as comma-separated host names without port numbers
 -zk_root <arg>                  name of parent znode in zookeeper
```

`wal`

The `wal` utility prints information about the contents of a specified WAL file. To get a list of all WAL files, use the HDFS command `hadoop fs -ls -R /hbase/WALs`. To run the `wal` utility, use the `bin/hbase wal` command. Run it without options to get usage information.

```
hbase wal
usage: WAL <filename...> [-h] [-j] [-p] [-r <arg>] [-s <arg>] [-w <arg>]
 -h,--help                Output help message
 -j,--json                Output JSON
 -p,--printvals           Print values
 -r,--region <arg>        Region to filter by. Pass encoded region name; e.g.
                          '9192caead6a5a20acb4454ffbc79fa14'
 -s,--sequence <arg>      Sequence to filter by. Pass sequence number.
 -w,--row <arg>           Row to filter by. Pass row name.
```

`hfile`

The `hfile` utility prints diagnostic information about a specified `hfile`, such as block headers or statistics. To get a list of all `hfiles`, use the HDFS command `hadoop fs -ls -R /hbase/data`. To run the `hfile` utility, use the `bin/hbase hfile` command. Run it without options to get usage information.

```
$ hbase hfile

usage: HFile [-a] [-b] [-e] [-f <arg> | -r <arg>] [-h] [-i] [-k] [-m] [-p]
       [-s] [-v] [-w <arg>]
 -a,--checkfamily            Enable family check
 -b,--printblocks            Print block index meta data
 -e,--printkey               Print keys
 -f,--file <arg>             File to scan. Pass full-path; e.g.
                             hdfs://a:9000/hbase/hbase:meta/12/34
 -h,--printblockheaders      Print block headers for each block.
 -i,--checkMobIntegrity      Print all cells whose mob files are missing
 -k,--checkrow               Enable row order check; looks for out-of-order
                             keys
 -m,--printmeta              Print meta data of file
 -p,--printkv                Print key/value pairs
 -r,--region <arg>           Region to scan. Pass region name; e.g.
                             'hbase:meta,,1'
 -s,--stats                  Print statistics
 -v,--verbose                Verbose output; emits file and meta data
                             delimiters
 -w,--seekToRow <arg>        Seek to this row and print all the kvs for this
                             row only
```

`hbck`

The `hbck` utility checks and optionally repairs errors in HFiles.

> **Warning:** Running hbck with any of the `-fix` or `-repair` commands is dangerous and can lead to data loss. Contact Cloudera support before running it.

To run hbck, use the bin/hbase hbck command. Run it with the -h option to get more usage information.

```
$ bin/hbase hbck -h

Usage: fsck [opts] {only tables}
 where [opts] are:
   -help Display help options (this)
   -details Display full report of all regions.
   -timelag <timeInSeconds>  Process only regions that  have not experienced any metadata updates
 in the last
           <timeInSeconds> seconds.
   -sleepBeforeRerun <timeInSeconds> Sleep this many seconds before checking if the fix worked if
 run with
                   -fix
   -summary Print only summary of the tables and status.
   -metaonly Only check the state of the hbase:meta table.
   -sidelineDir <hdfs://> HDFS path to backup existing meta.
   -boundaries Verify that regions boundaries are the same between META and store files.
   -exclusive Abort if another hbck is exclusive or fixing.
   -disableBalancer Disable the load balancer.

  Metadata Repair options: (expert features, use with caution!)
   -fix              Try to fix region assignments.  This is for backwards compatiblity
   -fixAssignments   Try to fix region assignments.  Replaces the old -fix
   -fixMeta          Try to fix meta problems.  This assumes HDFS region info is good.
   -noHdfsChecking   Don't load/check region info from HDFS. Assumes hbase:meta region info is
good. Won't
                     check/fix any HDFS issue, e.g. hole, orphan, or overlap
   -fixHdfsHoles     Try to fix region holes in hdfs.
   -fixHdfsOrphans   Try to fix region dirs with no .regioninfo file in hdfs
   -fixTableOrphans  Try to fix table dirs with no .tableinfo file in hdfs (online mode only)
   -fixHdfsOverlaps  Try to fix region overlaps in hdfs.
   -fixVersionFile   Try to fix missing hbase.version file in hdfs.
   -maxMerge <n>     When fixing region overlaps, allow at most <n> regions to merge. (n=5 by
default)
   -sidelineBigOverlaps  When fixing region overlaps, allow to sideline big overlaps
   -maxOverlapsToSideline <n>  When fixing region overlaps, allow at most <n> regions to sideline
 per group.
                     (n=2 by default)
   -fixSplitParents  Try to force offline split parents to be online.
   -ignorePreCheckPermission  ignore filesystem permission pre-check
   -fixReferenceFiles  Try to offline lingering reference store files
   -fixEmptyMetaCells  Try to fix hbase:meta entries not referencing any region (empty
REGIONINFO_QUALIFIER rows)

  Datafile Repair options: (expert features, use with caution!)
   -checkCorruptHFiles     Check all Hfiles by opening them to make sure they are valid
   -sidelineCorruptHFiles  Quarantine corrupted HFiles.  implies -checkCorruptHFiles

  Metadata Repair shortcuts
   -repair           Shortcut for -fixAssignments -fixMeta -fixHdfsHoles
                                  -fixHdfsOrphans -fixHdfsOverlaps -fixVersionFile
                                  -sidelineBigOverlaps -fixReferenceFiles -fixTableLocks
                                  -fixOrphanedTableZnodes
   -repairHoles      Shortcut for -fixAssignments -fixMeta -fixHdfsHoles

  Table lock options
   -fixTableLocks    Deletes table locks held for a long time (hbase.table.lock.expire.ms,
                     10min by default)

  Table Znode options
   -fixOrphanedTableZnodes    Set table state in ZNode to disabled if table does not exists

 Replication options
   -fixReplication   Deletes replication queues for removed peers
```

clean

After you have finished using a test or proof-of-concept cluster, the hbase clean utility can remove all HBase-related data from ZooKeeper and HDFS.

> **Warning:** The hbase clean command destroys data. Do not run it on production clusters, or unless you are absolutely sure you want to destroy the data.

To run the `hbase clean` utility, use the `bin/hbase clean` command. Run it with no options for usage information.

```
$ bin/hbase clean

Usage: hbase clean (--cleanZk|--cleanHdfs|--cleanAll)
Options:
        --cleanZk   cleans hbase related data from zookeeper.
        --cleanHdfs cleans hbase related data from hdfs.
        --cleanAll  cleans hbase related data from both zookeeper and hdfs.
```

Configuring HBase Garbage Collection

> **Warning:** Configuring the JVM garbage collection for HBase is an advanced operation. Incorrect configuration can have major performance implications for your cluster. Test any configuration changes carefully.

Garbage collection (memory cleanup) by the JVM can cause HBase clients to experience excessive latency. See Tuning Java Garbage Collection for HBase for a discussion of various garbage collection settings and their impacts on performance.

To tune the garbage collection settings, you pass the relevant parameters to the JVM.

Example configuration values are not recommendations and should not be considered as such. This is not the complete list of configuration options related to garbage collection. See the documentation for your JVM for details on these settings.

`-XX:+UseG1GC`

Use the 'G1' garbage collection algorithm. You can tune G1 garbage collection to provide a consistent pause time, which benefits long-term running Java processes such as HBase, NameNode, Solr, and ZooKeeper. For more information about tuning G1, see the Oracle documentation on tuning garbage collection.

`-XX:MaxGCPauseMillis=`*`value`*

The garbage collection pause time. Set this to the maximum amount of latency your cluster can tolerate while allowing as much garbage collection as possible.

`-XX:+ParallelRefProcEnabled`

Enable or disable parallel reference processing by using a + or - symbol before the parameter name.

`-XX:-ResizePLAB`

Enable or disable resizing of Promotion Local Allocation Buffers (PLABs) by using a + or - symbol before the parameter name.

`-XX:ParallelGCThreads=`*`value`*

The number of parallel garbage collection threads to run concurrently.

`-XX:G1NewSizePercent=`*`value`*

The percent of the heap to be used for garbage collection. If the value is too low, garbage collection is ineffective. If the value is too high, not enough heap is available for other uses by HBase.

If your cluster is managed by Cloudera Manager, follow the instructions in Configure HBase Garbage Collection Using Cloudera Manager on page 111. Otherwise, use Configure HBase Garbage Collection Using the Command Line on page 112.

Configure HBase Garbage Collection Using Cloudera Manager

Minimum Required Role: Full Administrator

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope** > **RegionServer**.
4. Select **Category** > **Advanced**.

5. Locate the **Java Configuration Options for HBase RegionServer** property or search for it by typing its name in the Search box.
6. Add or modify JVM configuration options.
7. Click **Save Changes** to commit the changes.
8. Restart the role.

Configure HBase Garbage Collection Using the Command Line

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

1. On each RegionServer, edit `conf/hbase-env.sh`.
2. Add or modify JVM configuration options on the line beginning with `HBASE_OPTS`.
3. Restart the RegionServer.

Disabling the `BoundedByteBufferPool`

HBase uses a `BoundedByteBufferPool` to avoid fragmenting the heap. The G1 garbage collector reduces the need to avoid fragmenting the heap in some cases. If you use the G1 garbage collector, you can disable the `BoundedByteBufferPool` in HBase in CDH 5.7 and higher. This can reduce the number of "old generation" items that need to be collected. This configuration is experimental.

To disable the `BoundedByteBufferPool`, set the `hbase.ipc.server.reservoir.enabled` property to `false`.

Disable the `BoundedByteBufferPool` Using Cloudera Manager

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select **Scope** > **RegionServer**.
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. Add the following XML:

```
<property>
  <name>hbase.ipc.server.reservoir.enabled</name>
  <value>false</value>
</property>
```

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

Disable the `BoundedByteBufferPool` Using the Command Line

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

1. On each RegionServer, edit `conf/hbase-site.xml`.
2. Add the following XML:

```
<property>
  <name>hbase.ipc.server.reservoir.enabled</name>
```

```
  <value>false</value>
</property>
```

3. Save your changes.
4. Restart the RegionServer.

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 Using Cloudera Manager on page 13.

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:

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

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

HBaseFsck (`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.

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 an 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.

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

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`

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 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'}}
```

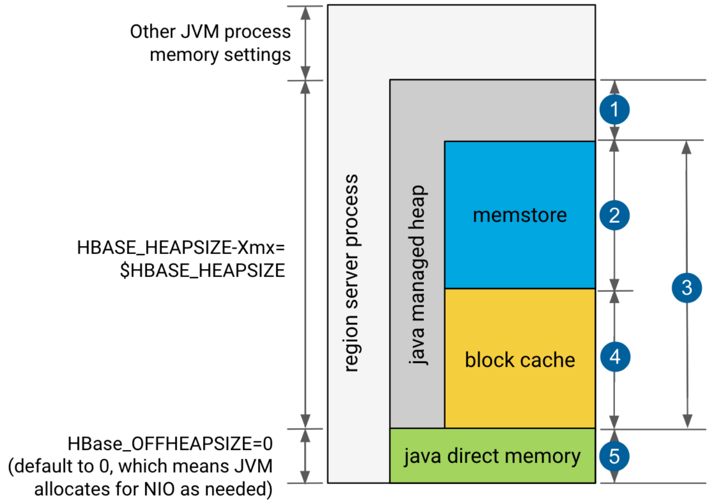
Configuring the off-heap `BucketCache`

This table summaries the important configuration properties for the `BucketCache`. To configure the `BucketCache`, see Configuring the off-heap BucketCache Using Cloudera Manager on page 121 or Configuring the off-heap BucketCache Using the Command Line on page 122. The table is followed by three diagrams that show the impacts of different blockcache settings.

Table 6: BucketCache Configuration Properties

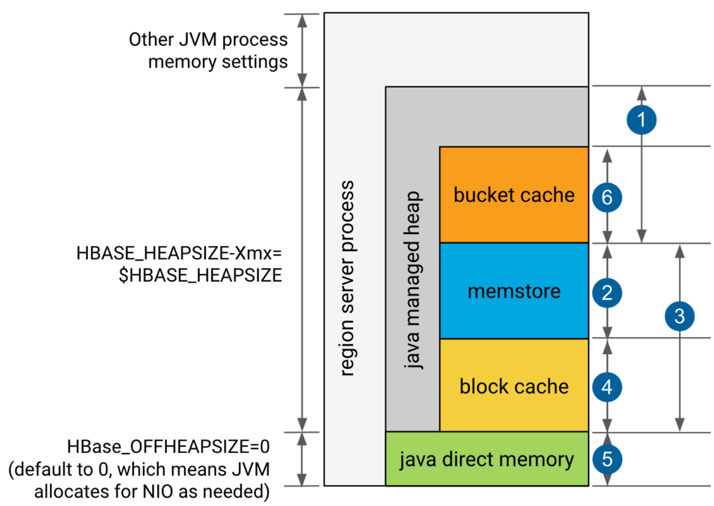
Property	Default	Description
hbase.bucketcache.combinedcache.enabled	`true`	When `BucketCache` is enabled, use it as a L2 cache for `LruBlockCache`. If set to true, indexes and Bloom filters are kept in the `LruBlockCache` and the data blocks are kept in the `BucketCache`

Property	Default	Description
`hbase.bucketcache.ioengine`	none (`BucketCache` is disabled by default)	Where to store the contents of the `BucketCache`. Either onheap or `file:`*`/path/to/file`*.
`hfile.block.cache.size`	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.
`hbase.bucketcache.size`	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. • 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
`hbase.bucketcache.bucket.sizes`	`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.**
`-XX:MaxDirectMemorySize`	`MaxDirectMemorySize = BucketCache + 1`	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: `MaxDirectMemorySize = BucketCache size + 1` GB for other features using direct memory, such as DFSClient. For example, if the BucketCache size is 8 GB, it will be `-XX:MaxDirectMemorySize=9G`.



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 LRUCache, L1 only block cache `hbase.bucketcache.ioengine=NULL`

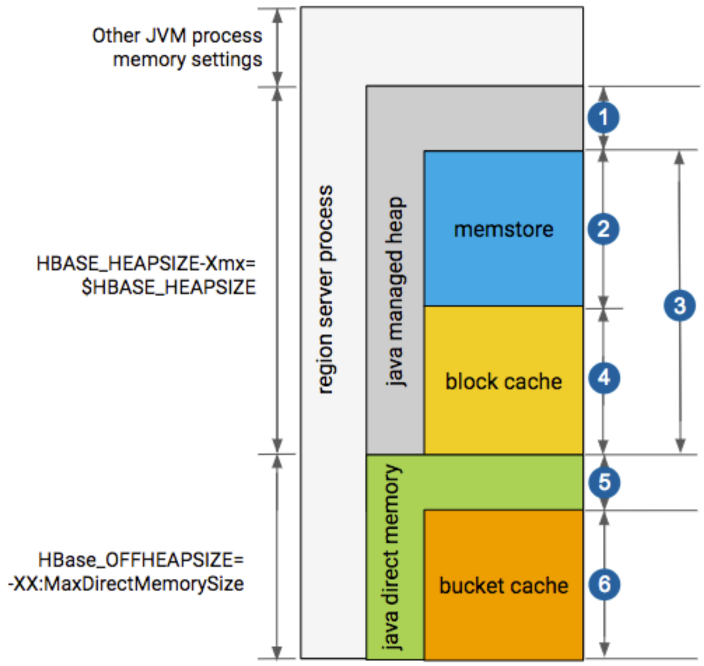


1. 20% minimum reserved for operations and rpc call queues
2. `hbase.regionserver.global.memstore.size`: default is 0.4, which means 40%
3. `hbase.regionserver.global.memstore.size` + `hfile.block.cache.size` ≤0.80, which means 80%
4. `hfile.block.cache.size`: default is 0.4, which means 40%
5. slack 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 LRUCache, 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 LRUCache, 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:MaxDirectMamorySize=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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

1. 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 DFSClient 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 6: BucketCache Configuration Properties on page 118 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.

Configuring the HBase Scanner Heartbeat

A *scanner heartbeat check* enforces a time limit on the execution of scan RPC requests. This helps prevent scans from taking too long and causing a timeout at the client.

When the server receives a scan RPC request, a time limit is calculated to be half of the smaller of two values: `hbase.client.scanner.timeout.period` and `hbase.rpc.timeout` (which both default to 60000 milliseconds, or one minute). When the time limit is reached, the server returns the results it has accumulated up to that point. This result set may be empty. If your usage pattern includes that scans will take longer than a minute, you can increase these values.

To make sure the timeout period is not too short, you can configure `hbase.cells.scanned.per.heartbeat.check` to a minimum number of cells that must be scanned before a timeout check occurs. The default value is 10000. A smaller value causes timeout checks to occur more often.

Configure the Scanner Heartbeat 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 **HBase or HBase Service-Wide**.
4. Select **Category** > **Main**.
5. Locate the **RPC Timeout** property or search for it by typing its name in the Search box.
6. Edit the property.
7. To modify the default values for `hbase.client.scanner.timeout.period` or `hbase.cells.scanned.per.heartbeat.check`, search for **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml**. Paste one or both of the following properties into the field and modify the values as needed.

```
<property>
  <name>hbase.client.scanner.timeout.period</name>
  <value>60000</value>
</property>
<property>
  <name>hbase.cells.scanned.per.heartbeat.check</name>
  <value>10000</value>
</property>
```

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

Configure the Scanner Heartbeat Using the Command Line

1. Edit `hbase-site.xml` and add the following properties, modifying the values as needed.

```
<property>
  <name>hbase.rpc.timeout</name>
  <value>60000</value>
</property>
<property>
  <name>hbase.client.scanner.timeout.period</name>
  <value>60000</value>
</property>
<property>
  <name>hbase.cells.scanned.per.heartbeat.check</name>
  <value>10000</value>
</property>
```

2. Distribute the modified `hbase-site.xml` to all your cluster hosts and restart the HBase master and RegionServer processes for the change to take effect.

Limiting the Speed of Compactions

You can limit the speed at which HBase compactions run, by configuring `hbase.regionserver.throughput.controller` and its related settings. The default controller is `org.apache.hadoop.hbase.regionserver.compactions.PressureAwareCompactionThroughputController`, which uses the following algorithm:

1. If compaction pressure is greater than 1.0, there is no speed limitation.
2. In off-peak hours, use a fixed throughput limitation, configured using `hbase.hstore.compaction.throughput.offpeak`, `hbase.offpeak.start.hour`, and `hbase.offpeak.end.hour`.
3. In normal hours, the max throughput is tuned between `hbase.hstore.compaction.throughput.higher.bound` and `hbase.hstore.compaction.throughput.lower.bound` (which default to 20 MB/sec and 10 MB/sec respectively), using the following formula, where `compactionPressure` is between 0.0 and 1.0. The `compactionPressure` refers to the number of store files that require compaction.

```
lower + (higer - lower) * compactionPressure
```

To disable compaction speed limits, set `hbase.regionserver.throughput.controller` to `org.apache.hadoop.hbase.regionserver.compactions.NoLimitCompactionThroughputController`.

Configure the Compaction Speed 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 **HBase or HBase Service-Wide**.
4. Search for **HBase Service Advanced Configuration Snippet (Safety Valve) for hbase-site.xml**. Paste the relevant properties into the field and modify the values as needed. See Configure the Compaction Speed Using the Command Line on page 125 for an explanation of the properties.
5. Click **Save Changes** to commit the changes.
6. Restart the role.
7. Restart the service.

Configure the Compaction Speed Using the Command Line

1. Edit `hbase-site.xml` and add the relevant properties, modifying the values as needed. Default values are shown. `hbase.offpeak.start.hour` and `hbase.offpeak.end.hour` have no default values; this configuration sets the off-peak hours from 20:00 (8 PM) to 6:00 (6 AM).

```
<property>
  <name>hbase.regionserver.throughput.controller</name>

<value>org.apache.hadoop.hbase.regionserver.compactions.PressureAwareCompactionThroughputController</value>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.higher.bound</name>
  <value>20971520</value>
  <description>The default is 20 MB/sec</description>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.lower.bound</name>
  <value>10485760</value>
  <description>The default is 10 MB/sec</description>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.offpeak</name>
  <value>9223372036854775807</value>
  <description>The default is Long.MAX_VALUE, which effectively means no
limitation</description>
</property>
<property>
  <name>hbase.offpeak.start.hour</name>
  <value>20</value>
  <value>When to begin using off-peak compaction settings, expressed as an integer
between 0 and 23.</value>
</property>
<property>
  <name>hbase.offpeak.end.hour</name>
  <value>6</value>
  <value>When to stop using off-peak compaction settings, expressed as an integer between
 0 and 23.</value>
</property>
<property>
  <name>hbase.hstore.compaction.throughput.tune.period</name>
  <value>60000</value>
  <description>
</property>
```

2. Distribute the modified `hbase-site.xml` to all your cluster hosts and restart the HBase master and RegionServer processes for the change to take effect.

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, the REST API, or the Thrift API.

- 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 128 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)}

# Specify a row prefix filter and another custom filter
hbase> scan 't1', {ROWPREFIXFILTER => 'row2',
                         FILTER => (QualifierFilter (>=, 'binary:xyz')) AND
(TimestampsFilter ( 123, 456))}

# 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:

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

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 125.

Dynamically Loading a Custom Filter

CDH 5.5 and higher adds (and enables by default) the ability to dynamically load a custom filter by adding a JAR with your filter to the directory specified by the `hbase.dynamic.jars.dir` property (which defaults to the `lib/` directory under the HBase root directory).

To disable automatic loading of dynamic JARs, set `hbase.use.dynamic.jars` to `false` in the advanced configuration snippet for `hbase-site.xml` if you use Cloudera Manager, or to `hbase-site.xml` otherwise.

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 the HBase Shell, the REST API, the Thrift 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 138.

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. Futher, 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 checks that the HBase tables can be upgraded, and then upgrades 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 true
  => 'NONE', BLOOMFILTER => 'ROW', REPLICATION_SCOPE
  => '0', VERSIONS => '1', COMPRESSION => 'NONE', MI
 N_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_C
 ELLS => 'false', BLOCKSIZE => '65536', IN_MEMORY =>
  'false', BLOCKCACHE => 'true'}, {NAME => 'cf2', DA
 TA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'ROW',
 REPLICATION_SCOPE => '0', VERSIONS => '1', COMPRESS
 ION => 'NONE', MIN_VERSIONS => '0', TTL => 'FOREVER
 ', KEEP_DELETED_CELLS => 'false', BLOCKSIZE => '655
 36', 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.RMProxy: 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`,

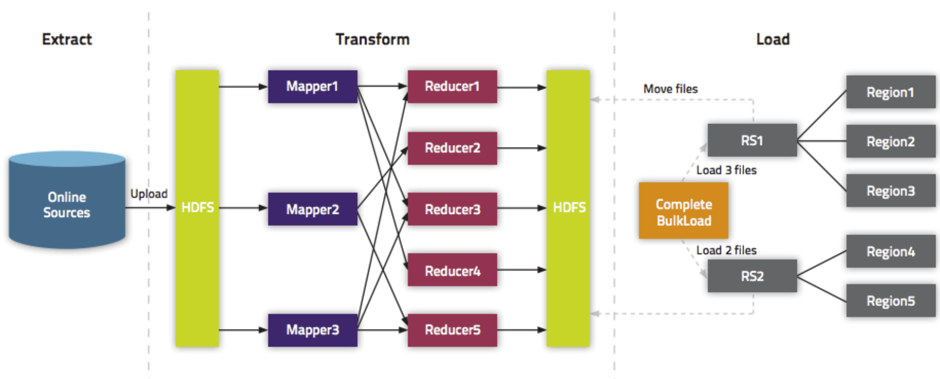
or a `Delete` as the `Value`. The Reducer is handled by HBase; configure it using HFileOutputFormat.configureIncrementalLoad() and it does the following:

- 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 the 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.

Using BulkLoad On A Secure Cluster

If you use security, HBase allows you to securely BulkLoad data into HBase. For a full explanation of how secure BulkLoad works, see HBase Transparent Encryption at Rest.

First, configure a `hbase.bulkload.staging.dir` which will be managed by HBase and whose subdirectories will be writable (but not readable) by HBase users. Next, add the `org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint` coprocessor to your configuration, so that users besides the `hbase` user can BulkLoad files into HBase. This functionality is available in CDH 5.5 and higher.

```
<property>
  <name>hbase.bulkload.staging.dir</name>
  <value>/tmp/hbase-staging</value>
</property>
<property>
  <name>hbase.coprocessor.region.classes</name>
  <value>org.apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint</value>
</property>
```

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. Replication is supported both from CDH 5 to CDH 6 and from CDH 6 to CDH 5, the source and destination cluster do not have to run the same major version of CDH.

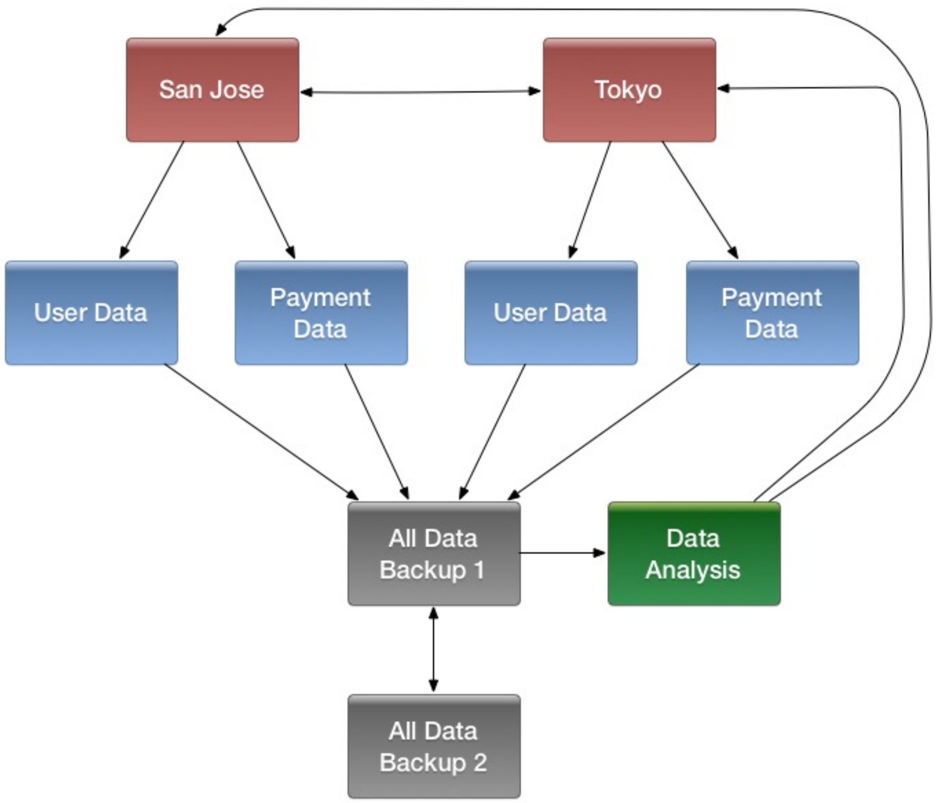
Cluster replication uses an active-push methodology. An HBase cluster can be a source (also called *active*, meaning that it writes new data), a destination (also called *passive*, meaning that it receives data using replication), or can fulfill both roles at once. Replication is asynchronous, and the 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 505 and [Configuring Secure HBase Replication](#). The following is a high-level overview of the steps to enable replication.

> **Important:** To run replication-related HBase comands, 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.
2. 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. See [Creating the Empty Table On the Destination Cluster](#) on page 511.
3. On the source cluster, enable replication in Cloudera Manager, or by setting `hbase.replication` to `true` in `hbase-site.xml`.

4. 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
```

5. 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.

6. 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'
```

7. 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
```

8. 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=timestamp1] [--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 511 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 ouses 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. T 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 iexample 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(linenumber).zfill(6)

    mutations = [
            Hbase.Mutation(column=messagecolumncf, value=line.strip()),
            Hbase.Mutation(column=linenumbercolumncf, value=encode(linenumber)),
            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 = ch1
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.ch1.type=memory
```

HBaseSink:

```
#Use the HBaseSink
host1.sinks.sink1.type = org.apache.flume.sink.hbase.HBaseSink
host1.sinks.sink1.channel = ch1
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 and Using the HBase REST API

You can use the HBase REST API to interact with HBase services, tables, and regions using HTTP endpoints.

Installing the REST Server

The HBase REST server is an optional component of HBase and is not installed by default.

Installing the REST Server Using Cloudera Manager

Minimum Required Role: Full Administrator

1. Click the **Clusters** tab.
2. Select **Clusters** > ***HBase***.
3. Click the **Instances** tab.
4. Click **Add Role Instance**.
5. Under **HBase REST Server**, click **Select Hosts**.
6. Select one or more hosts to serve the HBase Rest Server role. Click **Continue**.
7. Select the HBase Rest Server roles. Click **Actions For Selected** > **Start**.
8. To configure Kerberos authentication between REST clients and servers, see Configure Authentication for the HBase REST and Thrift Gateways.

Installing the REST Server Using the Command Line

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

Follow these instructions for each HBase host fulfilling the REST server role.

- To start the REST server as a foreground process, use the following command:

```
$ bin/hbase rest start
```

- To start the REST server as a background process, use the following command:

```
$ bin/hbase-daemon.sh start rest
```

- To use a different port than the default of 8080, use the `-p` option.
- To stop a running HBase REST server, use the following command:

```
$ bin/hbase-daemon.sh stop rest
```

- To configure Kerberos authentication between REST clients and servers, see Configure Authentication for the HBase REST and Thrift Gateways.

Using the REST API

The HBase REST server exposes endpoints that provide CRUD (create, read, update, delete) operations for each HBase process, as well as tables, regions, and namespaces. For a given endpoint, the HTTP verb controls the type of operation (create, read, update, or delete).

> **Note: `curl` Command Examples**
>
> The examples in these tables use the `curl` command, and follow these guidelines:
>
> - The HTTP verb is specified using the `-X` parameter.
> - For GET queries, the `Accept` header is set to `text/xml`, which indicates that the client (curl) expects to receive responses formatted in XML. You can set it to `text/json` to receive JSON responses instead.
> - For PUT, POST, and DELETE queries, the `Content-Type` header should be set only if data is also being sent with the `-d` parameter. If set, it should match the format of the data being sent, to enable the REST server to deserialize the data correctly.
>
> For more details about the `curl` command, see the documentation for the `curl` version that ships with your operating system.

These examples use port 20050, which is the default port for the HBase REST server when you use Cloudera Manager. If you use CDH without Cloudera Manager, the default port for the REST server is 8080.

Table 7: Cluster-Wide Endpoints

Endpoint	HTTP Verb	Description	Example
/version/cluster	GET	Version of HBase running on this cluster	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/version/cluster"`
/status/cluster	GET	Cluster status	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/status/cluster"`
/	GET	List of all nonsystem tables	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/"`

Table 8: Namespace Endpoints

Endpoint	HTTP Verb	Description	Example
/namespaces	GET	List all namespaces.	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/namespaces/"`
/namespaces/*namespace*	GET	Describe a specific namespace.	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/namespaces/special_ns"`
/namespaces/*namespace*	POST	Create a new namespace.	`curl -vi -X POST \ -H "Accept: text/xml" \`

Endpoint	HTTP Verb	Description	Example
			`"example.com:20550/namespaces/special_ns"`
/namespaces/*namespace*/tables	GET	List all tables in a specific namespace.	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/namespaces/special_ns/tables"`
/namespaces/*namespace*	PUT	Alter an existing namespace. Currently not used.	`curl -vi -X PUT \ -H "Accept: text/xml" \ "http://example.com:20550/namespaces/special_ns"`
/namespaces/*namespace*	DELETE	Delete a namespace. The namespace must be empty.	`curl -vi -X DELETE \ -H "Accept: text/xml" \ "example.com:20550/namespaces/special_ns"`

Table 9: Table Endpoints

Endpoint	HTTP Verb	Description	Example
/*table*/schema	GET	Describe the schema of the specified table.	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/users/schema"`
/*table*/schema	POST	Create a new table, or replace an existing table's	`curl -vi -X POST \ -H "Accept: text/xml" \`

Endpoint	HTTP Verb	Description	Example

		schema with the provided schema	`-H "Content-Type: text/xml" \`
			`-d '<?xml version="1.0" encoding="UTF-8"?><TableSchema name="users"><ColumnSchema name="cf" /></TableSchema>' \`
			`"http://example.com:20550/users/schema"`

/*table*/schema	UPDATE	Update an existing table with the provided schema fragment	`curl -vi -X PUT \`
			`-H "Accept: text/xml" \`
			`-H "Content-Type: text/xml" \`
			`-d '<?xml version="1.0" encoding="UTF-8"?><TableSchema name="users"><ColumnSchema name="cf" KEEP_DELETED_CELLS="true" /></TableSchema>' \`
			`"http://example.com:20550/users/schema"`

/*table*/schema	DELETE	Delete the table. You must use the `table/`schema endpoint, not just `table/`.	`curl -vi -X DELETE \`
			`-H "Accept: text/xml" \`
			`"http://example.com:20550/users/schema"`

/*table*/regions	GET	List the table regions.	`curl -vi -X GET \`
			`-H "Accept: text/xml" \`
			`"http://example.com:20550/users/regions"`

Table 10: Endpoints for `Get` Operations

Endpoint	HTTP Verb	Description	Example

/*table*/*row*/*column:qualifier*/*timestamp*	GET	Get the value of a single row. Values are Base-64 encoded.	Latest version:
			`curl -vi -X GET \`
			`-H "Accept: text/xml" \`
			`"http://example.com:20550/users/row1"`

Endpoint	HTTP Verb	Description	Example
			Specific timestamp: `curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/users/row1/cf:a/1458586888395"`

		Get the value of a single column. Values are Base-64 encoded.	Latest version:
			`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/users/row1/cf:a"`
			Specific version:
			`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/users/row1/cf:a/"`

/table/row/column:qualifier?v=number_of_versions		Multi-Get a specified number of versions of a given cell. Values are Base-64 encoded.	`curl -vi -X GET \ -H "Accept: text/xml" \ "http://example.com:20550/users/row1/cf:a?v=2"`

Table 11: Endpoints for Scan Operations

Endpoint	HTTP Verb	Description	Example
/*table*/scanner/	PUT	Get a Scanner object. Required by all other Scan operations. Adjust the batch parameter to the number of rows the scan should return in a batch. See the next example for adding filters to your Scanner. The scanner endpoint URL is returned as	`curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<Scanner batch="1"/>' \`

Endpoint	HTTP Verb	Description	Example
		the Location in the HTTP response. The other examples in this table assume that the Scanner endpoint is http://example.com:20550/users/scanner/145869072824375522207	"http:// example.com:20550/ users/scanner/"

/*table*/scanner/	PUT	To supply filters to the Scanner object or configure the Scanner in any other way, you can create a text file and add your filter to the file. For example, to return only rows for which keys start with u123 and use a batch size of 100:
		<Scanner batch="100">
		  <filter>
		    {
		      "type": "PrefixFilter",
		      "value": "u123"
		    }
		  </filter>
		</Scanner>
		Pass the file to the -d argument of the curl request.
			curl -vi -X PUT \\
			-H "Accept: text/xml" \\
			-H "Content-Type:text/xml" \\
			-d @filter.txt \\
			"http://example.com:20550/users/scanner/"

/*table*/scanner/*scanner_id*	GET	Get the next batch from the scanner. Cell values are byte-encoded. If the scanner	curl -vi -X GET \\ -H "Accept: text/

Endpoint	HTTP Verb	Description	Example
		is exhausted, HTTP status 204 is returned.	`xml" \ "http://example.com:20550/users/scanner/145869072824375522207"`
/table/scanner/*scanner_id*	DELETE	Deletes the scanner and frees the resources it was using.	`curl -vi -X DELETE \ -H "Accept: text/xml" \ "http://example.com:20550/users/scanner/145869072824375522207"`

Table 12: Endpoints for `Put` Operations

Endpoint	HTTP Verb	Description	Example
/table/row_key/	PUT	Write a row to a table. The row, column qualifier, and value must each be Base-64 encoded. To encode a string, you can use the `base64` command-line utility. To decode the string, use `base64 -d`. The payload is in the --data argument, so the `/users/fakerow` value is a placeholder. Insert multiple rows by adding them to the `<CellSet>` element. You can also save the data to be inserted to a file and pass it to the `-d` parameter with the syntax `-d @filename.txt`.	XML: `curl -vi -X PUT \ -H "Accept: text/xml" \ -H "Content-Type: text/xml" \ -d '<?xml version="1.0" encoding="UTF-8" standalone="yes"?><CellSet><Row key="cm93NQo="><Cell column="Y2Y6ZQo=">dmFsdWU1Cg==</Cell></Row></CellSet>' \ "http://example.com:20550/users/fakerow"` JSON: `curl -vi -X PUT \ -H "Accept: text/json" \ -H "Content-Type: text/json" \ -d '{"Row":[{"key":"cm93NQo=", "Cell": [{"column":"Y2Y6ZQo=", "$":"dmFsdWU1Cg=="}]}]}'' \`

Endpoint	HTTP Verb	Description	Example
			`"example.com:20550/users/fakerow"`

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

1. 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:

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

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 13: 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 configuration 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 13: HBase MOB Cache Properties on page 163 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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

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 13: HBase MOB Cache Properties on page 163 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.

Configuring the Storage Policy for the Write-Ahead Log (WAL)

In CDH 5.7.0 and higher, you can configure the preferred HDFS storage policy for HBase's write-ahead log (WAL) replicas. This feature allows you to tune HBase's use of SSDs to your available resources and the demands of your workload.

These instructions assume that you have followed the instructions in Configuring Storage Directories for DataNodes on page 175 and that your cluster has SSD storage available to HBase. If HDFS is not configured to use SSDs, these configuration changes will have no effect on HBase. The following policies are available:

- **NONE**: no preference about where the replicas are written.
- **ONE_SSD**: place one replica on SSD storage and the remaining replicas in default storage. This allows you to derive some benefit from SSD storage even if it is a scarce resource in your cluster.

> **Warning:** `ONE_SSD` mode has not been thoroughly tested with HBase and is not recommended.

- **ALL_SSD**: place all replicas on SSD storage.

Configuring the Storage Policy for WALs Using Cloudera Manager

Minimum Required Role: [Full Administrator]

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Search for the property **WAL HSM Storage Policy**.
4. Select your desired storage policy.
5. Save your changes. Restart all HBase roles.

Changes will take effect after the next major compaction.

Configuring the Storage Policy for WALs Using the Command Line

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

Paste the following XML into `hbase-site.xml`. Uncomment the <value> line that corresponds to your desired storage policy.

```
<property>
  <name>hbase.wal.storage.policy</name>
  <value>NONE</value>
  <!--<value>ONE_SSD</value>-->
  <!--<value>ALL_SSD</value>-->
</property>
```

> **Warning:** `ONE_SSD` mode has not been thoroughly tested with HBase and is not recommended.

Restart HBase. Changes will take effect for a given region during its next major compaction.

Exposing HBase Metrics to a Ganglia Server

[Ganglia] is a popular open-source monitoring framework. You can expose HBase metrics to a Ganglia instance so that Ganglia can detect potential problems with your HBase cluster.

Expose HBase Metrics to Ganglia Using Cloudera Manager

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

1. Go to the HBase service.
2. Click the **Configuration** tab.
3. Select the HBase Master or RegionServer role. To monitor both, configure each role as described in the rest of the procedure.

4. Select **Category** > **Metrics**.
5. Locate the **Hadoop Metrics2 Advanced Configuration Snippet (Safety Valve)** property or search for it by typing its name in the Search box.
6. Edit the property. Add the following, substituting the server information with your own.

```
hbase.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31
hbase.sink.ganglia.servers=<Ganglia server>:<port>
hbase.sink.ganglia.period=10
```

   If more than one role group applies to this configuration, edit the value for the appropriate role group. See Modifying Configuration Properties Using Cloudera Manager on page 13.

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

Expose HBase Metrics to Ganglia Using the Command Line

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

1. Edit `/etc/hbase/conf/hadoop-metrics2-hbase.properties` on the master or RegionServers you want to monitor, and add the following properties, substituting the server information with your own:

```
hbase.sink.ganglia.class=org.apache.hadoop.metrics2.sink.ganglia.GangliaSink31
hbase.sink.ganglia.servers=<Ganglia server>:<port>
hbase.sink.ganglia.period=10
```

2. Restart the master or RegionServer.

Using HashTable and SyncTable Tool

HashTable/SyncTable tool overview

HashTable/SyncTable is a two steps tool for synchronizing table data without copying all cells in a specified row key/time period range.

The HashTable/SyncTable tool can be used for partial or entire table data synchronization, under the same or remote cluster. Both the HashTable and the SyncTable step are implemented as a MapReduce job.

The first step, HashTable, creates hashed indexes for batch of cells on the source table and output those as results. The source table is the table whose state is copied to its counterpart.

The second step, SyncTable, scans the target table and calculates hash indexes for table cells. Then these hashes are compared to the HashTable step outputs. So, SyncTable scans and compares cells for diverging hashes and updating only the mismatching cells.

This results in less network traffic or data transfers than other methods, for example CopyTable, which can impact performance when large tables are synchronized on remote clusters.

Remote clusters are often deployed on different Kerberos Realms. SyncTable support cross realm authentication, allowing a SyncTable process running on the target cluster to connect to the source cluster and read both the HashTable output files and the given HBase table when performing the required comparisons.

HashTable/SyncTable tool configuration

You can configure the HashTable/SyncTable tool for your specific needs.

Using the batchsize option

You can define the amount of cell data for a given region that is hashed together in a single hash value using the `batchsize` option, which sets the batchsize property. Sizing this property has a direct impact on the synchronization efficiency. If the batch size is increased, larger chunks are hashed.

If only a few differences are expected between the two tables, using a bit larger batch size can be beneficial, as less scans are executed by mapper tasks of SyncTable.

However, if relatively frequent differences are expected between the tables, using a large batch size can cause frequent mismatches of hash values, as the probability of finding at least one mismatch in a batch is increased.

The following is an example of sizing this property:

```
$ hbase org.apache.hadoop.hbase.mapreduce.HashTable --batchsize=32000 --numhashfiles=50
 --starttime=1265875194289 --endtime=1265878794289 --families=cf2,cf3 TestTableA
/hashes/testTable
```

Creating a read only report

You can use the `dryrun` option in the second, SyncTable, step to create a read only report. It produces only COUNTERS indicating the differences between the two tables, but does not perform any actual changes. It can be used as an alternative of the VerifyReplication tool.

The following is an example of using this option:

```
 $ hbase org.apache.hadoop.hbase.mapreduce.SyncTable --dryrun=true
--sourcezkcluster=zk1.example.com,zk2.example.com,zk3.example.com:2181:/hbase
hdfs://nn:8020/hashes/testTable testTableA testTableB
```

Synchronize table data using HashTable/SyncTable

The HashTable/SyncTable tool can be used for partial or entire table data synchronization, under the same or remote cluster.

Prerequisites

- Ensure that all RegionServers/DataNodes on the source cluster is accessible by the NodeManagers on the target cluster where SyncTable job tasks will be running.
- In the case of secured clusters, the user on the target cluster who executes the SyncTable job must be able to do the following on the HDFS and HBase services of the source cluster:
  - Authenticate: for example, using centralized authentication.
  - Be authorized: having at least read permission.

Steps

1. Run `HashTable` on the source table cluster: `HashTable [options] <tablename> <outputpath>`.

   The following is an example, to has the TesTable in 32kB batches for a 1 hour window into 50 files:

```
 $ hbase org.apache.hadoop.hbase.mapreduce.HashTable --batchsize=32000 --numhashfiles=50
 --starttime=1265875194289 --endtime=1265878794289 --families=cf2,cf3 TestTableA
/hashes/testTable
```

   For more detailed information regarding HashTable options, use `hbase org.apache.hadoop.hbase.mapreduce.HashTable --help` .

2. Run `SyncTable` on the target cluster: `SyncTable [options] <sourcehashdir> <sourcetable> <targettable>`.

The following is an example, for a dry run SyncTable of tableA from a remote source cluster to a local tableB on the target cluster:

```
$ hbase org.apache.hadoop.hbase.mapreduce.SyncTable --dryrun=true
--sourcezkcluster=zk1.example.com,zk2.example.com,zk3.example.com:2181:/hbase
hdfs://nn:8020/hashes/testTable testTableA testTableB
```

For more detailed information regarding HashTable options, use `hbase org.apache.hadoop.hbase.mapreduce.SyncTable --help`.

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 372.

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**)

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 45.
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 171.

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 64 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. Clear 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 64 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.

Sizing NameNode Heap Memory

Each workload has a unique byte-distribution profile. Some workloads can use the default JVM settings for heap memory and garbage collection, but others require tuning. This topic provides guidance on sizing your NameNode JVM if the dynamic heap settings cause a bottleneck.

All Hadoop processes run on a Java Virtual Machine (JVM). The number of JVMs depend on your deployment mode:

- **Local** (or standalone) mode - There are no daemons and everything runs on a single JVM.
- **Pseudo-distributed** mode - Each daemon (such as the NameNode daemon) runs on its own JVM on a single host.
- **Distributed** mode - Each daemon runs on its own JVM across a cluster of hosts.

The standard NameNode configuration is one active (and primary) NameNode for the entire namespace and one Secondary NameNode for checkpoints (but not failover). A high- availability configuration replaces the Secondary NameNode with a Standby NameNode that prevents a single point of failure. Each NameNode uses its own JVM.

Environment Variables

HADOOP_HEAPSIZE sets the JVM heap size for *all* Hadoop project servers such as HDFS, YARN, and MapReduce. *HADOOP_HEAPSIZE* is an integer passed to the JVM as the maximum memory (Xmx) argument. For example:

```
HADOOP_HEAPSIZE=1024
```

HADOOP_NAMENODE_OPTS is specific to the NameNode and sets all JVM flags, which must be specified. *HADOOP_NAMENODE_OPTS* overrides the *HADOOP_HEAPSIZE* Xmx value for the NameNode. For example:

```
HADOOP_NAMENODE_OPTS=-Xms1024m -Xmx1024m -XX:+UseParNewGC -XX:+UseConcMarkSweepGC
-XX:CMSInitiatingOccupancyFraction=70 -XX:+CMSParallelRemarkEnabled
-XX:+PrintTenuringDistribution -XX:OnOutOfMemoryError={{AGENT_COMMON_DIR}}/killparent.sh
```

Both *HADOOP_NAMENODE_OPTS* and *HADOOP_HEAPSIZE* are stored in `/etc/hadoop/conf/hadoop-env.sh`.

Monitoring Heap Memory Usage

You can monitor your heap memory usage several ways:

- **Cloudera Manager:** Look at the NameNode chart for heap memory usage. If you need to build the chart from scratch, run:

```
select jvm_max_memory_mb, jvm_heap_used_mb where roleType="NameNode"
```

- **NameNode Web UI:** Scroll down to the Summary and look for "Heap Memory used."
- **Command line:** Generate a heap dump.

Files and Blocks

In HDFS, data and metadata are decoupled. Data files are split into block files that are stored, and replicated, on DataNodes across the cluster. The filesystem namespace tree and associated metadata are stored on the NameNode.

Namespace objects are file inodes and blocks that point to block files on the DataNodes. These namespace objects are stored as a file system image (fsimage) in the NameNode's memory and also persist locally. Updates to the metadata are written to an edit log. When the NameNode starts, or when a checkpoint is taken, the edits are applied, the log is cleared, and a new fsimage is created.

> **Important:** The NameNode keeps the entire namespace image in memory. The Secondary NameNode, on its own JVM, does the same when creating an image checkpoint.

On average, each file consumes 1.5 blocks of storage. That is, the average file is split into two block files—one that consumes the entire allocated block size and a second that consumes half of that. On the NameNode, this same average file requires three namespace objects—one file inode and two blocks.

Disk Space versus Namespace

The CDH default block size (*dfs.blocksize*) is set to 128 MB. Each namespace object on the NameNode consumes approximately 150 bytes.

On DataNodes, data files are measured by disk space consumed—the actual data length—and not necessarily the full block size. For example, a file that is 192 MB consumes 192 MB of disk space and *not* some integral multiple of the block size. Using the default block size of 128 MB, a file of 192 MB is split into two block files, one 128 MB file and one 64 MB file. On the NameNode, namespace objects are measured by the number of files and blocks. The same 192 MB file is represented by three namespace objects (1 file inode + 2 blocks) and consumes approximately 450 bytes of memory.

Large files split into fewer blocks generally consume less memory than small files that generate many blocks. One data file of 128 MB is represented by two namespace objects on the NameNode (1 file inode + 1 block) and consumes approximately 300 bytes of memory. By contrast, 128 files of 1 MB each are represented by 256 namespace objects (128 file inodes + 128 blocks) and consume approximately 38,400 bytes. The optimal split size, then, is some integral multiple of the block size, for memory management as well as data locality optimization.

By default, Cloudera Manager allocates a maximum heap space of 1 GB for every million blocks (but never less than 1 GB). How much memory you actually need depends on your workload, especially on the number of files, directories, and blocks generated in each namespace. If all of your files are split at the block size, you could allocate 1 GB for every million **files**. But given the historical average of 1.5 blocks per file (2 block objects), a more conservative estimate is 1 GB of memory for every million **blocks**.

> **Important:** Cloudera recommends 1 GB of NameNode heap space per million blocks to account for the namespace objects, necessary bookkeeping data structures, and the remote procedure call (RPC) workload. In practice, your heap requirements will likely be less than this conservative estimate.

Replication

The default block replication factor (*dfs.replication*) is three. **Replication affects disk space but not memory consumption.** Replication changes the amount of storage required for each block but not the number of blocks. If one block file on a DataNode, represented by one block on the NameNode, is replicated three times, the number of block files is tripled but not the number of blocks that represent them.

With replication off, one file of 192 MB consumes 192 MB of disk space and approximately 450 bytes of memory. If you have one million of these files, or 192 TB of data, you need 192 TB of disk space and, *without considering the RPC workload*, 450 MB of memory: (1 million inodes + 2 million blocks) * 150 bytes. With default replication on, you need 576 TB of disk space: (192 TB * 3) but the memory usage stay the same, 450 MB. When you account for bookkeeping and RPCs, and follow the recommendation of 1 GB of heap memory for every million blocks, a much safer estimate for this scenario is 2 GB of memory (with or without replication).

Examples

Example 1: Estimating NameNode Heap Memory Used

Alice, Bob, and Carl each have 1 GB (1024 MB) of data on disk, but sliced into differently sized files. Alice and Bob have files that are some integral of the block size and require the least memory. Carl does not and fills the heap with unnecessary namespace objects.

Alice: 1 x 1024 MB file

- 1 file inode
- 8 blocks (1024 MB / 128 MB)

Total = 9 objects * 150 bytes = **1,350 bytes** of heap memory

Bob: 8 x 128 MB files

- 8 file inodes
- 8 blocks

Total = 16 objects * 150 bytes = **2,400 bytes** of heap memory

Carl: 1,024 x 1 MB files

- 1,024 file inodes
- 1,024 blocks

Total = 2,048 objects * 150 bytes = **307,200 bytes** of heap memory

Example 2: Estimating NameNode Heap Memory Needed

In this example, memory is estimated by considering the capacity of a cluster. Values are rounded. Both clusters physically store 4800 TB, or approximately 36 million *block files* (at the default block size). Replication determines how many namespace blocks represent these block files.

Cluster A: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, **Replication=1**
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 1 = **128 MB** storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 128 MB = **36,000,000 blocks**

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster A needs 36 GB of maximum heap space.

Cluster B: 200 hosts of 24 TB each = 4800 TB.

- Blocksize=128 MB, **Replication=3**
- Cluster capacity in MB: 200 * 24,000,000 MB = 4,800,000,000 MB (4800 TB)
- Disk space needed per block: 128 MB per block * 3 = **384 MB** storage per block
- Cluster capacity in blocks: 4,800,000,000 MB / 384 MB = **12,000,000 blocks**

At capacity, with the recommended allocation of 1 GB of memory per million blocks, Cluster B needs 12 GB of maximum heap space.

Both Cluster A and Cluster B store the same number of *block files*. In Cluster A, however, each block file is unique and represented by one block on the NameNode; in Cluster B, only one-third are unique and two-thirds are replicas.

Backing Up and Restoring NameNode Metadata

This topic describes the steps for backing up and restoring NameNode metadata.

Backing Up NameNode Metadata

This section describes how to back up NameNode metadata.

1. Make a single backup of the VERSION file. This does not need to be backed up regularly as it does not change, but it is important since it contains the clusterID, along with other details.
2. Use the following command to back up the NameNode metadata. It automatically determines the active NameNode, retrieves the current fsimage, and places it in the defined backup_dir.

```
$ hdfs dfsadmin -fetchImage backup_dir
```

On startup, the NameNode process reads the `fsimage` file and commits it to memory. If the JournalNodes are up and running, and there are edit files present, any edits newer than the `fsimage` are also applied. If the JournalNodes are unavailable, it is possible to lose any data transferred in the interim.

Restoring NameNode Metadata

This section describes how to restore NameNode metadata. If both the NameNode and the secondary NameNode were to suddenly go offline, you can restore the NameNode by doing the following:

1. Add a new host to your Hadoop cluster.
2. Add the NameNode role to the host. Make sure it has the same hostname as the original NameNode.
3. Create a directory path for the NameNode `name.dir` (for example, `/dfs/nn/current`), ensuring that the permissions are set correctly.
4. Copy the `VERSION` and latest `fsimage` file to the `/dfs/nn/current` directory.
5. Run the following command to create the md5 file for the fsimage.

```
$ md5sum fsimage > fsimage.md5
```

6. Start the NameNode process.

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. The default replication factor for HDFS is three. That is, three copies of data are maintained at all times.

> **Important:** Cloudera recommends that you do not configure a lower replication factor when you have at least three DataNodes. A lower replication factor may lead to data loss.

Within a cluster, DataNodes should be uniform. If they are not uniform, issues can occur. For example, DataNodes with less memory fill up more quickly than DataNodes with more memory, which can result in job failures.

How NameNode Manages Blocks on a Failed DataNode

A DataNode is considered dead after a set period without any heartbeats (10.5 minutes by default). When this happens, the NameNode performs the following actions to maintain the configured replication factor (3x replication by default):

1. The NameNode determines which blocks were on the failed DataNode.
2. The 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.

Keep the following in mind when working with dead DataNodes:

- If the DataNode failed due to a disk failure, follow the procedure in Replacing a Disk on a DataNode Host on page 175 or Performing Disk Hot Swap for DataNodes on page 177 to bring a repaired DataNode back online. If a DataNode failed to heartbeat for other reasons, they need to be recommissioned to be added back to the cluster. For more information, see Recommissioning Hosts on page 71.
- If a DataNode rejoins the cluster, there is the possibility that there will be a surplus of replicas for blocks that were on that DataNode. The NameNode will randomly remove excess replicas while adhering to Rack-Awareness policies.

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 177.

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.

Configuring 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. To specify the storage type for HDFS heterogenous storage, add the storage type, surrounded by brackets, at the front of the path. For example: `[SSD]/data/example_dir/`

   If more than one role group applies to this configuration, edit the value for the appropriate role group. See Modifying Configuration Properties Using Cloudera Manager on page 13.

5. Click **Save Changes** to commit the changes.
6. Restart the DataNode.

> **Important:** You must restart the DataNodes for heterogenous storage configuration changes to take effect.

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 Using Cloudera Manager on page 13.

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 Using Cloudera Manager on page 13.

6. Click **Save Changes** to commit the changes.
7. Restart the role.

Configuring Storage Balancing for DataNodes Using the Command Line

This section applies to unmanaged deployments *without* Cloudera Manager. 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 /` command to validate the health of HDFS.

Performing Disk Hot Swap for DataNodes Using the Command Line

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

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-availabilty 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 46 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 169.

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 to use short-circuit local reads.

> **Note:** If you are configuring short-circuit reads on a CDH cluster that uses the DSSD Storage Appliance, see DSSD D5 and Short-Circuit 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 Using Cloudera Manager on page 13.
6. Click **Save Changes** to commit the changes.

Configuring Short-Circuit Reads Using the Command Line

Important:

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

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. When you delete a file in HDFS, the file is not immediately expelled from HDFS. Deleted files are first moved to the `/user/<username>/.Trash/Current` directory, with their original filesystem path being preserved. After a user-configurable period of time (`fs.trash.interval`), a process known as trash checkpointing renames the `Current` directory to the current timestamp, that is, `/user/<username>/.Trash/<timestamp>`. The checkpointing process also checks the rest of the `.Trash` directory for any existing timestamp directories and removes them from HDFS permanently. You can restore files and directories in the trash 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.

Trash Behavior with HDFS Transparent Encryption Enabled

Starting with CDH 5.7.1, you can delete files or directories that are part of an HDFS encryption zone. As is evident from the procedure described above, moving and renaming files or directories is an important part of trash handling in HDFS. However, currently HDFS transparent encryption only supports renames *within* an encryption zone. To accommodate this, HDFS creates a local `.Trash` directory every time a new encryption zone is created. For example, when you create an encryption zone, `/enc_zone`, HDFS will also create the `/enc_zone/.Trash/` sub-directory. Files deleted from enc_zone are moved to `/enc_zone/.Trash/<username>/Current/`. After the checkpoint, the `Current` directory is renamed to the current timestamp, `/enc_zone/.Trash/<username>/<timestamp>`.

If you delete the entire encryption zone, it will be moved to the `.Trash` directory under the user's home directory, `/users/<username>/.Trash/Current/enc_zone`. Trash checkpointing will occur only after the entire zone has been moved to `/users/<username>/.Trash`. However, if the user's home directory is already part of an encryption zone, then attempting to delete an encryption zone will fail because you cannot move or rename directories across encryption zones.

If you have upgraded your cluster to CDH 5.7.1 (or higher), and you have an encryption zone that was created before the upgrade, create the `.Trash` directory using the `-provisionTrash` option as follows:

```
$ hdfs crypto -provisionTrash -path /enc_zone
```

In **CDH 5.7.0 and lower**, HDFS does not automatically create the `.Trash` directory when an encryption zone is created. However, you can use the following commands to manually create the `.Trash` directory within an encryption zone. Make sure you run the commands as an admin user.

```
$ hdfs dfs -mkdir /enc_zone/.Trash
$ hdfs dfs -chmod 1777 /enc_zone/.Trash
```

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 clear 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 Using Cloudera Manager on page 13.

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 Using Cloudera Manager](#) on page 13.

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 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 Using Cloudera Manager](#) on page 13.

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

Configuring Concurrent Moves

The property `dfs.datanode.balance.max.concurrent.moves` sets the maximum number of threads used by the DataNode balancer for pending moves. It is a throttling mechanism to prevent the balancer from taking too many resources from the DataNode and interfering with normal cluster operations. Increasing the value allows the balancing process to complete more quickly, decreasing the value allows rebalancing to complete more slowly, but is less likely to compete for resources with other tasks on the DataNode. To use this property, you need to set the value on both the DataNode and the Balancer.

- To configure the Datanode:
  - Go to the HDFS service.
  - Click the **Configuration** tab.
  - Search for **DataNode Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml**.

- Add the following code to the configuration field, for example, setting the value to 50.

```
<property>
  <name>dfs.datanode.balance.max.concurrent.moves</name>
  <value>50</value>
</property>
```

- Restart the DataNode.

- To configure the Balancer:

1. Go to the HDFS service.
2. Click the **Configuration** tab.
3. Search for **Balancer Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml**.
4. Add the following code to the configuration field, for example, setting the value to 50.

```
<property>
  <name>dfs.datanode.balance.max.concurrent.moves</name>
  <value>50</value>
</property>
```

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:

- Follow these command-line instructions on systems that do not use Cloudera Manager.
- This information applies specifically to CDH 5.9.x. See [Cloudera Documentation]() for information specific to other releases.

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 `setBalancerBandwidth` and other HDFS command-line options, see the dfsadmin documentation.
- The property `dfs.datanode.balance.max.concurrent.moves` sets the maximum number of threads used by the DataNode balancer for pending moves. It is a throttling mechanism to prevent the balancer from taking too many resources from the DataNode and interfering with normal cluster operations. Increasing the value allows the balancing process to complete more quickly, decreasing the value allows rebalancing to complete more slowly, but is less likely to compete for resources with other tasks on the DataNode. Adjust the value of this property in the `/etc/hadoop/[service name]/hdfs-site.xml` configuration file.

```
<property>
  <name>dfs.datanode.balance.max.concurrent.moves</name>
  <value>50</value>
</property>
```

- 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: `swebhdfs://<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

Configure the HttpFS Service to work with the load balancer you configured for the service:

1. In the **Cloudera Manager Admin Console**, navigate to **Cluster** > **<HDFS service>**.
2. On the **Configuration** tab, search for the following property:

```
HttpFS Load Balancer
```

3. Enter the hostname and port for the load balancer in the following format:

```
<hostname>:<port>
```

4. Save the changes.

> **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 higher and CDH 5.0.1 or higher, 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 lower than Cloudera Manager 5.0.0 and CDH 5.0.0.
- The nfs-utils OS package is required for a client to mount the NFS export and to run commands such as showmount from the NFS Gateway.
- 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 Using Cloudera Manager on page 13.

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 Using Cloudera Manager on page 13.

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:

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

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.
- 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 190, 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>
```

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

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 375 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 `umount` 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

- Cloudera recommends 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 communicates with DataNodes and instructs them to cache specific blocks in off-heap caches.

Centralized and explicit caching has several advantages:

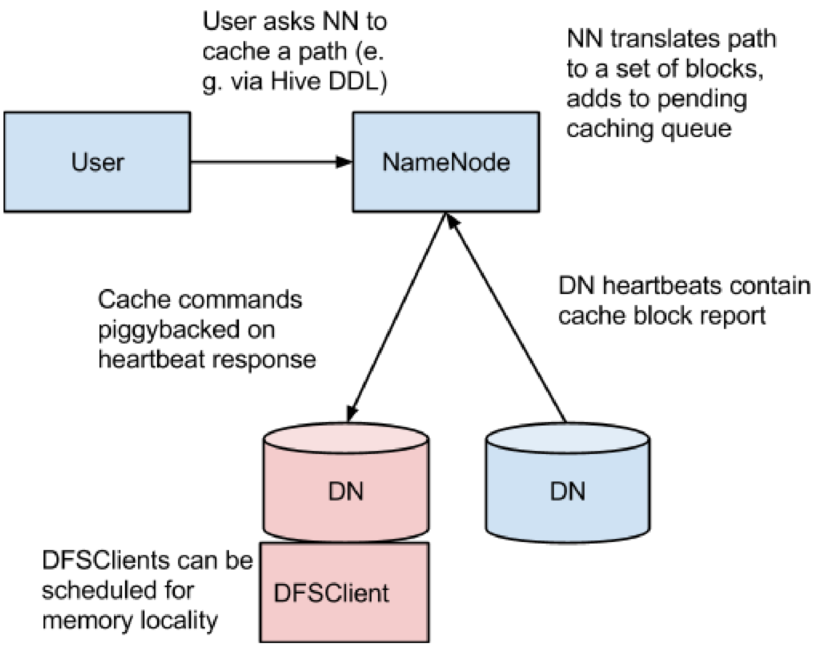
- Frequently used data is pinned in memory. This is important when the size of the working set exceeds the size of main memory, which is common for many HDFS workloads.
- Cluster memory is optimized because you can pin `m` of n block replicas, saving `n-m` memory. Before centralized pinning, repeated reads of a block caused all n replicas to be pulled into each DataNode buffer cache.
- Tasks are co-located with cached block replicas, improving read performance. Because the NameNode manages DataNode caches, applications can query the set of cached block locations when making task placement decisions.
- Clients can use the zero-copy read API, and incur almost no overhead, because each DataNode does a checksum verification of cached data only once.

Use Cases

Centralized cache management is best used for files that are accessed repeatedly. For example, a fact table in Hive that is often used in `JOIN` clauses is a good candidate for caching. Caching the input of an annual reporting query is probably less useful, as the historical data might be read only once.

Centralized cache management is also useful for mixed workloads with performance service-level agreements (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 assigns caching to DataNodes. Rescans can also be triggered by user actions such as adding or removing a cache directive or removing a cache pool.

Currently, blocks that are under construction, corrupt, or otherwise incomplete are not cached. If a cache directive covers a symlink, the symlink target is not cached. Caching is currently done on a per-file basis (and not at the block-level).

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 are cached.

Directives have parameters, such as the cache replication factor and expiration time. 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. 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 not used.

Cache pools are also used for resource management. Pools can enforce a maximum `limit` that restricts the aggregate number of bytes that can be cached by directives in the pool. Normally, the sum of the pool limits roughly equals 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 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 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 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 uses 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 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 30000, which is 30 seconds.
- `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`, 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, adjust the DataNode `ulimit -l` value. Usually, this value is configured in `/etc/security/limits.conf`; but varies depending on your operating system and distribution.

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. It is typical for `ulimit -l` to output the memory lock limit in KB, but `dfs.datanode.max.locked.memory` must be specified in bytes.

Configuring Proxy Users to Access HDFS

Hadoop allows you to configure proxy users to submit jobs or access HDFS on behalf of other users; this is called ***impersonation***. When you enable impersonation, any jobs submitted using a proxy are executed with the impersonated user's existing privilege levels rather than those of a superuser (such as `hdfs`). Because all proxy users are configured in one location, `core-site.xml`, Hadoop administrators to implement centralized access control.

To configure proxy users, set the `hadoop.proxyuser.<proxy_user>.hosts`, `hadoop.proxyuser.<proxy_group>.groups` and `hadoop.proxyuser.<proxy_user>.users` in `core-site.xml` properties.

For example, to allow user `alice` to impersonate a user belonging to `group_a` and `group_b`, set `hadoop.proxyuser.<proxy_group>.groups` as follows:

```
<property>
  <name>hadoop.proxyuser.alice.groups</name>
  <value>group_a,group_b</value>
</property>
```

To limit the hosts from which impersonated connections are allowed, use `hadoop.proxyuser.<proxy_user>.hosts`. For example, to allow user `alice` impersonated connections only from `host_a` and `host_b`:

```
<property>
   <name>hadoop.proxyuser.alice.hosts</name>
   <value>host_a,host_b</value>
</property>
```

If the configuration properties described are not present, impersonation is not allowed and connections will fail.

For looser restrictions, use a wildcard (`*`) to allow impersonation from any host and of any user. For example, to allow user bob to impersonate any user belonging to any group, and from any host, set the properties as follows:

```
<property>
  <name>hadoop.proxyuser.bob.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.bob.groups</name>
  <value>*</value>
</property>
```

The `hadoop.proxyuser.<proxy_user>.hosts` property also accepts comma-separated lists of IP addresses, IP address ranges in CIDR format, or host names. For example, to allow user `kate` access from hosts in the range `10.222.0.0-15` and `10.113.221.221`, to impersonate `user_a` and `user_b`, set the proxy user properties as follows:

```
<property>
     <name>hadoop.proxyuser.super.hosts</name>
     <value>10.222.0.0/16,10.113.221.221</value>
</property>
<property>
     <name>hadoop.proxyuser.super.users</name>
     <value>user1,user2</value>
</property>
```

Proxy Users for Kerberos-Enabled Clusters

For secure clusters, the proxy users must have Kerberos credentials to impersonate another user.

Proxy users cannot use delegation tokens. If a user is allowed to add its own delegation token to the proxy user UGI, it also allows the proxy user to connect to the service with the privileges of the original user.

If a superuser wants to give a delegation token to a proxy-user UGI, for example, `alice`, the superuser must first impersonate `alice`, get a delegation token for `alice`, and add it to the UGI for the newly created proxy UGI. This way, the delegation token has its owner set to `alice`.

Using CDH with Isilon Storage

EMC Isilon is a storage service with a distributed filesystem that can 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 CDH. For information about tasks performed on Isilon OneFS, see the Dell EMC Community's Isilon Info Hub.

Supported Versions

For Cloudera and Isilon compatibility information, see the product compatibility matrix for Product Compatibility for EMC Isilon.

> **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 Cloudera and Isilon Implementation

1. Create an Isilon access zone with HDFS support. For example:

```
/ifs/your-access-zone/hdfs
```

> **Note:** The above is an example; the HDFS root directory does not have to begin with `ifs` or end with `hdfs`.

2. CDH includes a default setting in `hdfs-site.xml` to support the WRT checksum type on datanodes. You must run the following OneFS command to support WRT checksum:

```
isi hdfs --checksum-type=crc32
```

3. Create two directories to 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`. For 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`. For 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
```

4. Create the service-specific users, groups, or directories for each CDH service you plan to use. Create the directories in the access zone you have created.

> **Note:** Many of the values 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 in the access zone you have created; for example, `/ifs/ your-access-zone /hdfs/`.

- ZooKeeper: nothing required.
- HBase
  - Create the `hbase` group with `hbase` user.
  - Create the root directory for HBase. For example:

```
hdfs_root_directory/hbase hbase:hbase 755
```

- YARN (MR2)
  - Create the `mapred` group with `mapred` user.
  - Create the `cmjobuser` user and add it to the hadoop group. For example:

```
isi-cloudera-1# isi auth users create cmjobuser --zone subnet1
isi-cloudera-1# isi auth users modify cmjobuser --add-group hadoop --zone subnet1
```

  - Create the `cloudera-scm` user and add it to the `supergroup` group. For example:

```
isi-cloudera-1# isi auth users create cloudera-scm --zone subnet1
isi-cloudera-1# isi auth users modify cloudera-scm --add-group supergroup --zone subnet1
```

  - Create `history` directory for YARN. For example:

```
hdfs_root_directory/user/history mapred:hadoop 777
```

  - Create the remote application log directory for YARN. For example:

```
hdfs_root_directory/tmp/logs mapred:hadoop 775
```

  - Create the `cmjobuser` directory for YARN. For example:

```
hdfs_root_directory/user/cmjobuser cmjobuser:hadoop 775
```

  - Create the `cloudera-scm` directory for YARN. For example:

```
hdfs_root_directory/user/cloudera-scm cloudera-scm:supergroup 775
```

  - Create the `tmp/cmYarnContainerMetrics` directory. For example:

```
hdfs_root_directory/tmp/cmYarnContainerMetrics cmjobuser:supergroup 775
```

  - Create the `tmp/cmYarnContainerMetricsAggregate` directory. For example:

```
hdfs_root_directory/tmp/cmYarnContainerMetricsAggregate cloudera-scm:supergroup 775
```

- Oozie
  - Create the `oozie` group with `oozie` user.

  - Create the user directory for Oozie. For example:

```
hdfs_root_directory/user/oozie oozie:oozie 775
```

- Flume
  - Create the `flume` group with `flume` user.
  - Create the user directory for Flume. For example:

```
hdfs_root_directory/user/flume flume:flume 775
```

- Hive
  - Create the `hive` group with `hive` user.
  - Create the user directory for Hive. For example:

```
hdfs_root_directory/user/hive hive:hive 775
```

  - Create the warehouse directory for Hive. For example:

```
hdfs_root_directory/user/hive/warehouse hive:hive 1777
```

  - Create a temporary directory for Hive. For example:

```
hdfs_root_directory/tmp/hive hive:supergroup 777
```

- Solr
  - Create the `solr` group with `solr` user.
  - Create the data directory for Solr. For example:

```
hdfs_root_directory/solr solr:solr 775
```

- Sqoop
  - Create the sqoop group with sqoop2 user.
  - Create the user directory for Sqoop. For example:

```
hdfs_root_directory/user/sqoop2 sqoop2:sqoop 775
```

- Hue
  - Create the hue group with hue user.
  - Create `sample` group with `sample` user.
  - Create the user directory for Hue. For example:

```
hdfs_root_directory/user/hue hue:hue 775
```

- Spark
  - Create the `spark` group with `spark` user.
  - Create the user directory for Spark. For example:

```
hdfs_root_directory/user/spark spark:spark 751
```

- Create the application history directory for Spark. For example:

```
hdfs_root_directory/user/spark/applicationHistory spark:spark 1777
```

5. Map the `hdfs` user to `root` on the Isilon service. For example:

```
isiloncluster1-1# isi zone zones modify
    --user-mapping-rules="hdfs=>root" --zone zone1
isiloncluster1-1# isi services isi_hdfs_d disable ; isi services isi_hdfs_d enable
The service 'isi_hdfs_d' has been disabled.
The service 'isi_hdfs_d' has been enabled.
```

If you are using Cloudera Manager, also map the `cloudera-scm` user to `root` on the Isilon service. For example:

```
isiloncluster1-1# isi zone zones modify --user-mapping-rules="cloudera-scm=>root" --zone
 zone1
isiloncluster1-1# isi services isi_hdfs_d disable ; isi services isi_hdfs_d enable
The service 'isi_hdfs_d' has been disabled.
The service 'isi_hdfs_d' has been enabled.
```

6. Create the following proxy users for the Flume, Impala, Hive, Hue, and Oozie services:

Service Proxy User	Users to Add as Members
Flume	• `impala` • `hive` • `oozie` • `yarn`
Impala	• `hive` • `oozie` • `yarn` • `hue`
Hive	• `oozie` • `yarn` • `hue` • `impala`
Hue	• `oozie` • `yarn` • `impala` • `hive`
Oozie	• `hive` • `hue` • `yarn`

Create the proxy users on the Isilon system by running the following command as root:

```
isi hdfs proxyusers create username --add-user username1 --add-user username2 ...
--zone access_zone
```

For example, to create proxy users for Hue:

```
isi hdfs proxyusers create hue --add-user oozie --add-user yarn --add-user impala
--add-user
      hive --zone subnet1
```

Note:

- If the CDH logs for your cluster contain messages saying that a user is not allowed to impersonate a certain user, add that user to the proxy users list using the following command:

```
isi hdfs proxyusers modify username --add-user newuser --zone access
zone
```

- If you have groups of users who need to run jobs from a service, add that user group to the proxy users using the following command:

```
isi hdfs proxyusers modify username --add-group group --zone access
zone
```

Once the users, groups, and directories are created in Isilon OneFS, you can install Cloudera Manager.

Installing Cloudera Manager with Isilon

To install Cloudera Manager, follow the instructions in Installing Cloudera Manager and CDH.

- 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 - Installation Using Cloudera Manager Parcels or Packages describes configuring external databases for Cloudera Manager and CDH storage.

If you choose parcel installation on the **Cluster Installation** screen, the installation wizard points to the latest parcels of CDH available.

On the installation wizard **Cluster Setup** page, click **Custom Services**, and select the services to install in the cluster. Be sure to select the **Isilon** service; do not select the **HDFS** service, and do not check **Include Cloudera Navigator** at the bottom of the **Cluster Setup** page. On the **Role Assignments** page, 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 insecure Cloudera Manager cluster as described above in Installing Cloudera Manager with Isilon on page 204.
2. Follow the Isilon documentation to enable Kerberos for your access zone: Cloudera CDH with Isilon and Active Directory Kerberos Implementation. This includes adding a Kerberos authentication provider to your Isilon access zone.
3. Follow the instructions in Configuring Authentication in Cloudera Manager to configure a secure cluster with Kerberos.

Upgrading a Cluster with Isilon

To upgrade CDH and Cloudera Manager in a cluster that uses Isilon:

1. If required, upgrade OneFS to a version compatible with the version of CDH to which you are upgrading. See the product compatibility matrix for Product Compatibility for EMC Isilon. For OneFS upgrade instructions, see the EMC Isilon documentation.
2. (Optional) Upgrade Cloudera Manager. See Cloudera Upgrade.

   The Cloudera Manager minor version must always be *equal to or greater than* the CDH minor version because older versions of Cloudera Manager may not support features in newer versions of CDH. For example, if you want to upgrade to CDH 5.4.8 you must first upgrade to Cloudera Manager 5.4 or higher.

3. Upgrade CDH. See Upgrading CDH and Managed Services Using 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 versions 5.4.4 through 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 Using CDH with Isilon Storage on page 199.

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.

Configuring Replication with Kerberos and Isilon

If you plan to use replication between clusters that use Isilon storage and that also have enabled Kerberos, do the following:

1. Create a custom Kerberos Keytab and Kerberos principal that the replication jobs use to authenticate to storage and other CDH services. See Configuring Authentication.
2. In Cloudera Manager, select **Administration** > **Settings**.
3. Search for and enter values for the following properties:
   - **Custom Kerberos Keytab Location** – Enter the location of the Custom Kerberos Keytab.
   - **Custom Kerberos Principal Name** – Enter the principal name to use for replication between secure clusters.
4. When you create a replication schedule, enter the **Custom Kerberos Principal Name** in the **Run As Username** field. See Configuring Replication of HDFS Data on page 480 and Configuring Replication of Hive Data on page 493.
5. Ensure that both the source and destination clusters have the same set of users and groups. When you set ownership of files (or when maintaining ownership), if a user or group does not exist, the chown command fails on Isilon. See Performance and Scalability Limitations on page 480
6. Cloudera recommends that you do not select the **Replicate Impala Metadata** option for Hive replication schedules. If you need to use this feature, create a custom principal of the form `hdfs/`*`hostname@realm`* or `impala/`*`hostname@realm`*.
7. Add the following property and value to the **HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** and **Cluster-wide Advanced Configuration Snippet (Safety Valve) for core-site.xml** properties:

```
hadoop.security.token.service.use_ip = false
```

If the replication MapReduce job fails with the an error similar to the following:

```
java.io.IOException: Failed on local exception: java.io.IOException:
  org.apache.hadoop.security.AccessControlException:
  Client cannot authenticate via:[TOKEN, KERBEROS];
  Host Details : local host is: "foo.mycompany.com/172.1.2.3";
  destination host is: "myisilon-1.mycompany.com":8020;
```

Set the Isilon cluster-wide time-to-live setting to a higher value on the *destination* cluster for the replication: Note that higher values may affect load balancing in the Isilon cluster by causing workloads to be less distributed. A value of 60 is a good starting point. For example:

```
isi networks modify pool subnet4:nn4 --ttl=60
```

You can view the settings for a subnet with a command similar to the following:

```
isi networks list pools --subnet subnet3 -v
```

Configuring Heterogeneous Storage in HDFS

CDH supports a variety of storage types in the Hadoop Distributed File System (HDFS). Earlier releases of CDH used a single (or homogeneous) storage model. Now you can choose which storage type to assign to each DataNode Data Directory. Specifying a storage type allows you to optimize your data usage and lower your costs, based on your data usage frequency. This topic describes these storage types and how to configure CDH to use them.

Overview

Each DataNode in a cluster is configured with a set of data directories. You can configure each data directory with a storage type. The storage policy dictates which storage types to use when storing the file or directory.

Some reasons to consider using different types of storage are:

- You have datasets with temporal locality (for example, time-series data). The latest data can be loaded initially into SSD for improved performance, then migrated out to disk as it ages.

- You need to move cold data to denser archival storage because the data will rarely be accessed and archival storage is much cheaper. This could be done with simple age-out policies: for example, moving data older than six months to archival storage.

Storage Types

The storage type identifies the underlying storage media. HDFS supports the following storage types:

- ARCHIVE - Archival storage is for very dense storage and is useful for rarely accessed data. This storage type is typically cheaper per TB than normal hard disks.
- DISK - Hard disk drives are relatively inexpensive and provide sequential I/O performance. This is the default storage type.
- SSD - Solid state drives are useful for storing hot data and I/O-intensive applications.
- RAM_DISK - This special in-memory storage type is used to accelerate low-durability, single-replica writes.

When you add the DataNode Data Directory, you can specify which type of storage it uses, by prefixing the path with the storage type, in brackets. If you do not specify a storage type, it is assumed to be `DISK`. See Adding Storage Directories on page 175.

Storage Policies

A storage policy contains information that describes the type of storage to use. This policy also defines the fallback storage type if the primary type is out of space or out of quota. If a target storage type is not available, HDFS attempts to place replicas on the default storage type.

Each storage policy consists of a policy ID, a policy name, a list of storage types, a list of fallback storage types for file creation, and a list of fallback storage types for replication.

HDFS has six preconfigured storage policies.

- Hot - All replicas are stored on DISK.
- Cold - All replicas are stored ARCHIVE.
- Warm - One replica is stored on DISK and the others are stored on ARCHIVE.
- All_SSD - All replicas are stored on SSD.
- One_SSD - One replica is stored on SSD and the others are stored on DISK.

> **Note:** You cannot create your own storage policy. You must use one of the six pre-configured policies. HDFS clients such as HBase may support different storage policies.

Setting Up SSD Storage Using Cloudera Manager

1. Set up your cluster normally, but customize your DataNodes with the [ssd] prefix for data directories. Adding [ssd] can also be done after initial setup (which requires an extra HDFS restart).
2. Stop HBase.
3. Using the HDFS client, move `/hbase` to `/hbase_backup`.
4. Re-create `/hbase` using the Cloudera Manager command in the HBase service (this ensures that proper permissions are used).
5. Using the HDFS client, set the storage policy for `/hbase` to be SSD only.
6. Use the DistCp to copy `/hbase_backup` to `/hbase`.

   ```
   hadoop distcp /hbase_backup /hbase
   ```

7. Start HBase.

Setting a Storage Policy for HDFS

Setting a Storage Policy for HDFS Using Cloudera Manager

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

To set a storage policy on a DataNode Data Directory using Cloudera Manager, perform the following tasks:

1. Check the **HDFS Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** to be sure that `dfs.storage.policy.enabled` has not been changed from its default value of `true`.
2. Specify the storage types for each DataNode Data Directory that is not a standard disk, by adding the storage type in brackets at the beginning of the directory path. For example:

```
[SSD]/dfs/dn1
[DISK]/dfs/dn2
[ARCHIVE]/dfs/dn3
```

3. Open a terminal session on any HDFS host. Run the following hdfs command for each path on which you want to set a storage policy:

```
$ hdfs storagepolicies -setStoragePolicy -path <path> -policy <policy>
path_to_file_or_directory -policy policy_name
```

4. To move the data to the appropriate storage based on the current storage policy, use the `mover` utility, from any HDFS host. Use `mover -h` to get a list of available options. To migrate all data at once (this may take a long time), you can set the path to `/`.

```
$ hdfs mover -p <path>
```

Note: Quotas are enforced at the time you set the storage policy or when writing the file, not when quotas are changed. The Mover tool does not recognize quota violations. It only verifies that a file is stored on the storage types specified in its policy. For more information about quotas, see Setting HDFS Quotas on page 191.

Setting a Storage Policy for HDFS Using the Command Line

Important:

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

To set a storage policy on a file or directory, perform the following tasks:

1. Make sure the `dfs.storage.policy.enabled` property (in the `conf/hdfs-site.xml` file) is set to `true`. This is the default setting.
2. Make sure the storage locations are tagged with their storage types. The default storage type is `DISK` if the directory does not contain a storage type tag. Add the storage type to the `dfs.datanode.dir` property in `conf/hdfs-site.xml`.

```
<property>
  <name>dfs.datanode.dir</name>
  <value>
   [DISK]file:///grid/dn/disk0,[SSD]file:///grid/dn/ssd0,[RAM_DISK]file:///grid/dn/ram0,

   [ARCHIVE]file:///grid/dn/archive0
  </value>
</property>
```

3. Set the storage policy for the HDFS path. Enter the following command on any HDFS host:

```
$ hdfs storagepolicies -setStoragePolicy -path <path> -policy <policy>
path_to_file_or_directory -policy policy_name
```

4. To move the data to the appropriate storage based on the current storage policy, use the `mover` utility, from any HDFS host. Use `mover -h` to get a list of available options. To migrate all data at once (this may take a long time), you can set the path to `/`.

```
$ hdfs mover -p <path>
```

> **Note:** Quotas are enforced at the time you set the storage policy or when writing the file, not when quotas are changed. The Mover tool does not recognize quota violations. It only verifies that a file is stored on the storage types specified in its policy. For more information about quotas, see Setting HDFS Quotas on page 191.

Managing Storage Policies

- To get the storage policy for a specific file or directory on a DataNode, use the following command, which is available using the command line on a any HDFS host.

```
$ hdfs storagepolicies -getStoragePolicy -path <path>path_to_policy
```

- To list all policies on a DataNode, enter the following command:

```
$ hdfs storagepolicies -listPolicies
```

- To reset a storage policy, follow the steps used in Setting a Storage Policy for HDFS on page 207.

Migrating Existing Data

To move the data to the appropriate storage based on the current storage policy, use the `mover` utility, from any HDFS host. Use `mover -h` to get a list of available options. To migrate all data at once (this may take a long time), you can set the path to `/`.

```
$ hdfs mover -p <path>
```

> **Note:** Quotas are enforced at the time you set the storage policy or when writing the file, not when quotas are changed. The Mover tool does not recognize quota violations. It only verifies that a file is stored on the storage types specified in its policy. For more information about quotas, see Setting HDFS Quotas on page 191.

Managing Hive

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

Hive Roles

Hive is implemented in three roles:

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

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

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

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

For information about configuring a remote Hive metastore database with Cloudera Manager, see Cloudera Manager and Managed Service Datastores. To configure high availability for the Hive metastore, see Hive Metastore High Availability on page 427.

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

Hive Execution Engines

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

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

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

To determine the current setting, run

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

- **Cloudera Manager** (*Affects all queries, not recommended*).

  1. Go to the Hive service.
  2. Click the **Configuration** tab.
  3. Search for "execution".
  4. Set the **Default Execution Engine** property to MapReduce or Spark. The default is MapReduce.
  5. Click **Save Changes** to commit the changes.
  6. Return to the Home page by clicking the Cloudera Manager logo.
  7. Click ⏻ to invoke the cluster restart wizard.
  8. Click **Restart Stale Services**.
  9. Click **Restart Now**.
  10. Click **Finish**.

Managing Hive Using Cloudera Manager

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

How Hive Configurations are Propagated to Hive Clients

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

Considerations When Upgrading Cloudera Manager

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

When upgrading from Cloudera Manager versions before 4.5, Cloudera Manager automatically creates new Hive services to capture the previous implicit Hive dependency from Hue and Impala. Your previous services continue to function without impact. If Hue was using a Hive metastore backed by a Derby database, the newly created Hive

Metastore Server also uses Derby. Because Derby does not allow concurrent connections, Hue continues to work, but the new Hive Metastore Server does not run. The failure is harmless (because nothing uses this new Hive Metastore Server at this point) and intentional, to preserve the set of cluster functionality as it was before upgrade. Cloudera discourages the use of a Derby-backed Hive metastore due to its limitations and recommends switching to a different supported database.

Cloudera Manager provides a Hive configuration option to bypass the Hive Metastore Server. When this configuration is enabled, Hive clients, Hue, and Impala connect directly to the Hive metastore database. Prior to Cloudera Manager 4.5, Hue and Impala connected directly to the Hive metastore database, so the bypass mode is enabled by default when upgrading to Cloudera Manager 4.5 and higher. This ensures that the upgrade does not disrupt your existing setup. You should plan to disable the bypass mode, especially when using CDH 4.2 and higher. Using the Hive Metastore Server is the recommended configuration, and the WebHCat Server role requires the Hive Metastore Server to *not* be bypassed. To disable bypass mode, see Disabling Bypass Mode on page 211.

Cloudera Manager 4.5 and higher also supports HiveServer2 with CDH 4.2. In CDH 4, HiveServer2 is not added by default, but can be added as a new role under the Hive service (see Role Instances on page 55). In CDH 5, HiveServer2 is a mandatory role.

Disabling Bypass Mode

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

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

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Select **Scope** > ***HIVE service_name*** **(Service-Wide)**
4. Select **Category** > **Advanced**.
5. Clear the **Bypass Hive Metastore Server** property.
6. Click **Save Changes** to commit the changes.
7. Re-deploy Hive client configurations.
8. Restart Hive and any Hue or Impala services configured to use that Hive service.

Running Hive on Spark

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

Configuring Hive on Spark

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

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

- Configure the Hive client to use the Spark execution engine as described in Hive Execution Engines on page 210.
- Identify the Spark service that Hive uses. Cloudera Manager automatically sets this to the configured MapReduce or YARN service and the configured Spark service. See Configuring the Hive Dependency on a Spark Service on page 211.

Configuring the Hive Dependency on a Spark Service

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

1. Go to the Hive service.
2. Click the **Configuration** tab.
3. Search for the **Spark On YARN Service**. To configure the Spark service, select the Spark service name. To remove the dependency, select **none**.
4. Click **Save Changes** to commit the changes.

5. Go to the Spark service.
6. Add a Spark gateway role to the host running HiveServer2.
7. Return to the Home page by clicking the Cloudera Manager logo.
8. Click ⏻ to invoke the cluster restart wizard.
9. Click **Restart Stale Services**.
10. Click **Restart Now**.
11. Click **Finish**.
12. In the Hive client, configure the Spark execution engine.

Configuring Hive on Spark for Performance

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

For information on configuring Hive on Spark for performance, see Tuning Hive on Spark on page 292.

Troubleshooting Hive on Spark

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

Symptom

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

Cause

The query waits for YARN containers to initialize.

Solution

No action required. Subsequent queries will be faster.

Exception in HiveServer2 log and HiveServer2 is down

Symptom

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

Cause

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

Solution

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

Out-of-memory error

Symptom

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

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

Cause

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

Solution

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

Spark applications stay alive forever

Symptom

Cluster resources are consumed by Spark applications.

Cause

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

Solution

Manually terminate the Hive on Spark applications:

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

HiveServer2 Web UI

The HiveServer2 web UI provides access to Hive configuration settings, local logs, metrics, and information about active sessions and queries. The HiveServer2 web UI is enabled in newly created clusters running CDH 5.7 and higher, and those using Kerberos are configured for SPNEGO. Clusters upgraded from a previous CDH version must be configured to enable the web UI; see HiveServer2 Web UI Configuration on page 213.

Accessing the HiveServer2 Web UI

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

The following information is displayed:

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

HiveServer2 Web UI Configuration

For managed deployments, configure the HiveServer2 web UI in Cloudera Manager. See Configuring the HiverServer2 Web UI in Cloudera Manager on page 214.

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

Configurable Properties

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

```
hive.server2.webui.max.threads=50
hive.server2.webui.host=0.0.0.0
hive.server2.webui.port=10002
hive.server2.webui.use.ssl=false
hive.server2.webui.keystore.path=""
hive.server2.webui.keystore.password=""
hive.server2.webui.max.historic.queries=25
hive.server2.webui.use.spnego=false
```

```
hive.server2.webui.spnego.keytab=""
hive.server2.webui.spnego.principal=<dynamically sets special string, _HOST, as
hive.server2.webui.host or host name>
```

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

Configuring the HiverServer2 Web UI in Cloudera Manager

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

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

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

1. Go to the **Hive** service.
2. Click the **Configuration** tab.
3. Select **Scope** > **HiveServer2**.
4. Search for "webui".
5. Locate the properties you want to set and enter your preferred values.
6. Click **Save Changes** to commit the changes.
7. Select **Actions** > **Restart** and when done, click **Close**.
8. Click **HiveServer2 Web UI** to view your changes.

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

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

Hive Table Statistics

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

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

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

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

Managing User-Defined Functions (UDFs) with HiveServer2

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

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

Blacklist for Built-in UDFs

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

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

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

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

Creating Permanent Functions

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

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

3. In the Cloudera Manager Admin Console, go to the Hive service.
4. Click the **Configuration** tab.
5. Expand the **Hive (Service-Wide)** scope.
6. Click the **Advanced** category.
7. Configure the **Hive Auxiliary JARs Directory** property with the HiveServer2 host path and the Hive Metastore host path from Step 2, for example `/opt/local/hive/lib/`. Setting this property overwrites `hive.aux.jars.path`, even if this variable has been previously set in the HiveServer2 advanced configuration snippet.
8. Click **Save Changes**. The JARs are added to `HIVE_AUX_JARS_PATH` environment variable.
9. Redeploy the Hive client configuration.
   a. In the Cloudera Manager Admin Console, go to the Hive service.
   b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
   c. Click **Deploy Client Configuration**.
10. Restart the Hive service.
11. ***With Sentry enabled*** - Grant privileges on the JAR files to the roles that require access. Log in to Beeline as user `hive` and use the Hive SQL GRANT statement to do so. For example:

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

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

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

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

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

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

Creating Temporary Functions

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

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

2. In the Cloudera Manager Admin Console, go to the Hive service.
3. Click the **Configuration** tab.
4. Expand the **Hive (Service-Wide)** scope.
5. Click the **Advanced** category.
6. Configure the **Hive Auxiliary JARs Directory** property with the HiveServer2 host path and the Hive Metastore host path from Step 1, for example `/opt/local/hive/lib/`. Setting this property overwrites `hive.aux.jars.path`, even if this variable has been previously set in the HiveServer2 advanced configuration snippet.
7. Click **Save Changes**. The JARs are added to `HIVE_AUX_JARS_PATH` environment variable.
8. Redeploy the Hive client configuration.
   a. In the Cloudera Manager Admin Console, go to the Hive service.
   b. From the **Actions** menu at the top right of the service page, select **Deploy Client Configuration**.
   c. Click **Deploy Client Configuration**.
9. Restart the Hive service.
10. Run the `CREATE TEMPORARY FUNCTION` command. For example:

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

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

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

Creating Permanent Functions

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

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

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

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

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

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

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

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

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

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

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

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

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

Creating Temporary Functions

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

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

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

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

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

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

Update Existing HiveServer2 User-Defined Functions (UDFs)

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

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

```
# more drop_func.hql

drop function my_udf;
```

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

```
# more drop_func.sh

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

Run the bash script:

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

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

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

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

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

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

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

```
 # more create_func.hql

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

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

```
# more create_func.sh

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

Then run the bash script:

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

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

Configuring Transient Hive ETL Jobs to Use the Amazon S3 Filesystem

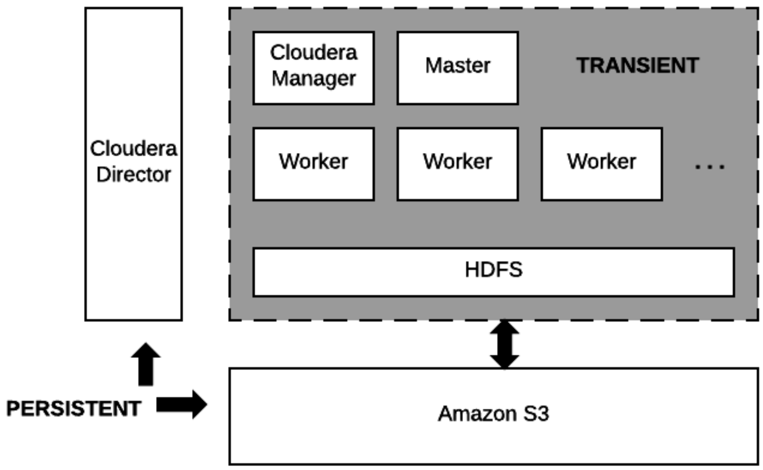
Hive is a popular choice for batch extract-transform-load (ETL) jobs such as cleaning, serializing, deserializing, and transforming data. In on-premise deployments, ETL jobs operate on data stored in a permanent Hadoop cluster that runs HDFS on local disks. However, ETL jobs are frequently transient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

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

About Transient Jobs

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

Transient Jobs Hosted on Amazon S3



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

Configuring and Running Jobs on Transient Clusters

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

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

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

Configuring AWS Settings

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

Best Practices
Networking

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

Data Access

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

AWS Placement Groups

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

Install Cloudera Director

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

Create the Cluster Configuration File

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

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

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

Testing the Cluster Configuration File

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

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

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

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

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

Prepare the CDH AMIs

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

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

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

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

```
    }
...
}
```

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

Prepare the Job Wrapper Script

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

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

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

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

Log Collection

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

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

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

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

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

Prepare the End-to-End Job Submission Script

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

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

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

Schedule the Recurring Job

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

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 in Cloudera Manager to add and configure a new Hue service instance.

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

   to the right of the cluster name.
2. Select **Add a Service**.
3. Select the **Hue** service and click **Continue**.
4. Select the row with the Hue dependencies required for your cluster. For more information, see Hue Dependencies.
5. Click **Continue** to accept the default role assignments; or click the gray field below each role to open the hosts dialog, customize assignments, and click **OK** to save.

   If a drop down menu displays (indicating that all hosts apply), select **All Hosts**, or else click **Custom** to display the hosts dialog. Click **OK** to accept custom assignments.

   The wizard evaluates host hardware configurations to determine the best hosts for each role. All worker roles are automatically assigned to the same set of hosts as the HDFS DataNode. You can reassign if necessary. Specify hostnames by IP address, rack name, or by range:

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

6. Select **Use Custom Databases** for production clusters and input values for database *hostname*, *type*, *name*, *username*, and *password*.
7. Click **Test Connection**, and when green, click **Continue**. Cloudera Manager starts the Hue service.
8. Click **Continue** and **Finish**.
9. If your cluster uses Kerberos, Cloudera Manager *automatically* adds a **Hue Kerberos Ticket Renewer** role to each host where you assigned the Hue Server role instance. 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**)

Roles are functions that comprise a service and role instances must be assigned to one or more hosts. You can easily assign roles to hosts in Cloudera Manager.

1. Go to the **Hue** service.
2. Click the **Instances** tab.
3. Click the **Add Role Instances** button.
4. Click **Continue** to accept the default role assignments; or click the gray field below each role to open the hosts dialog, customize assignments, and click **OK** to save.

   If a drop down menu displays (indicating that all hosts apply), select **All Hosts**, or else click **Custom** to display the hosts dialog. Click **OK** to accept custom assignments.

The wizard evaluates host hardware configurations to determine the best hosts for each role. All worker roles are automatically assigned to the same set of hosts as the HDFS DataNode. You can reassign if necessary. Specify hostnames by IP address, rack name, or by range:

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

5. If your cluster uses Kerberos, you must *manually* add the **Hue Kerberos Ticket Renewer** role to each host where you assigned the Hue Server role instance. Cloudera Manager throws a validation error if the new Hue Server role does not have a colocated KT Renewer role. 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 Using Cloudera Manager on page 391 for detailed instructions.

To configure the Hue service itself for high availability, see Hue High Availability on page 433.

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. Clear 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 Using Cloudera Manager on page 13.
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 and click the **Configuration** tab.
2. Filter by **Scope** > **Hue (Service-Wide)** and **Category** > **Main**.
3. Select each ***service name*** **Service** property to set the dependency. Select **none** to remove the dependency.
4. Click **Save Changes** to commit the changes.
5. Restart the Hue service.

Enabling the Sqoop 2 Application

If you are upgrading Cloudera Manager from version 4.6 or lower, you must 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 the proxy, HBase Thrift Server, which forwards commands to HBase. Because Hue stands between the Thrift server and the user, all HBase operations appear to come from the Hue user and not the actual user who is logged on. In a Keberos cluster, you can enable impersonation so that operations appear to come from the actual user.

1. Logon to Cloudera Manager.
2. Add the HBase Thrift Server role:
   a. Go to the **HBase** service and click the **Instances** tab.
   b. Click the button, **Add Role Instances**.
   c. Click **Select hosts** under HBase Thrift Server.
   d. Click anywhere in host row to add the purple icon, "HBTS," under Existing Roles.
   e. Click **OK** and **Continue**.
   f. Check the box by your new HBase Thrift Server and select **Actions for Selected** > **Start**.
3. If you have a Kerberos cluster, enable impersonation. If your cluster does not have Kerberos or TLS enabled, skip to step 6..

   > **Note:** Enabling impersonation requires that you grant Hbase permissions to each individual user. Otherwise, grant all HBase permissions to the Hue user.

   a. Click the HBase **Configuration** tab.
   b. Filter by **Scope** > **Service-Wide** and **Category** > **Security**.
   c. Set the property, **HBase Thrift Authentication** (hbase.thrift.security.qop), to one of the following values:
      - `auth-conf`: authentication, integrity and confidentiality checking
      - `auth-int`: authentication and integrity checking
      - `auth`: authentication only
   d. Filter by **Scope** > **Service-Wide** and **Category** > **Main**.
   e. Check the **Service-Wide** box for **Enable HBase Thrift Http Server** (hbase.regionserver.thrift.http) and **Enable HBase Thrift Proxy Users** (hbase.thrift.support.proxyuser).
   f. Click **Save Changes**.
4. If you have a Kerberos cluster with doAs and force principal names to lower case, be sure to exclude the HTTP principal:
   a. Go to the **HDFS** service.
   b. Filter by **Scope** > **HDFS (Service-Wide)** and **Category** > **Security**.
   c. Search on **Additional Rules to Map Kerberos Principals to Short Names** (`auth_to_local`) and add two HTTP rules above your existing rules:

```
# Exclude HTTP
RULE:[1:$1@$0](HTTP@\QEXAMPLE.COM\E$)s/@\Q.EXAMPLE.COM\E$//
RULE:[2:$1@$0](HTTP@\QEXAMPLE.COM\E$)s/@\Q.EXAMPLE.COM\E$//

# Force to Lower Case
RULE:[1:$1@$0](.*@\QEXAMPLE.COM\E$)s/@\Q.EXAMPLE.COM\E$///L
RULE:[2:$1@$0](.*@\QEXAMPLE.COM\E$)s/@\Q.EXAMPLE.COM\E$///L
```

   d. Click **Save Changes**.

e. Select **Actions** > **Deploy Client Configuration**.
f. Select **Cluster** > **Actions** > **Rolling Restart**, check the boxes for HDFS, HBase, and Hue and click **Rolling Restart**.

5. Enable TLS/SSL for the HBase Thrift Server:

a. Filter by **Scope** > **HBase Thrift Server** and **Category** > **Security**.
b. Set the TLS/SSL properties according to your cluster configuration:

Property	Description
Enable TLS/SSL for HBase Thrift Server over HTTP	Encrypt communication between clients and HBase Thrift Server over HTTP using Transport Layer Security (TLS).
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore File Location	Path to the TLS/SSL keystore file (in JKS format) with the TLS/SSL server certificate and private key. Used when HBase Thrift Server over HTTP acts as a TLS/SSL server.
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore File Password	Password for the HBase Thrift Server JKS keystore file.
HBase Thrift Server over HTTP TLS/SSL Server JKS Keystore Key Password	Password that protects the private key contained in the JKS keystore used when HBase Thrift Server over HTTP acts as a TLS/SSL server.

c. Click **Save Changes** to commit the changes.
d. Restart the **HBase** service.

6. Configure Hue to point to the Thrift Server and to a valid HBase configuration directory:

a. Go to the **Hue** service and click the **Configuration** tab.
b. Filter by **Scope** > **All** and **Category** > **Main**.
c. Set the property, **HBase Service**, to the service for which you enabled the Thrift Server role (if you have more than one HBase service instance).
d. Set the property, **HBase Thrift Server**, to the Thrift Server role for Hue to use.
e. Filter by **Category** > **Advanced**.
f. Edit the property, **Hue Service Advanced Configuration Snippet (Safety Valve) for hue_safety_valve.ini**, by adding a valid HBase configuration directory as follows:

```
[hbase]
hbase_conf_dir={{HBASE_CONF_DIR}}
```

g. 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 248 for detailed instructions.

Using an External Database for Hue

See the Hue Guide for information on configuring Hue with an external database:

- Connect Hue to MySQL or MariaDB
- Connect Hue to PostgreSQL
- Connect Hue to Oracle with Client Parcel
- Connect Hue to Oracle with Client Package

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 229
- Configuring Impala to Work with ODBC on page 230
- Configuring Impala to Work with JDBC on page 232

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 Apache Impala - Interactive SQL.

For information on features that support Impala resource management see Admission Control and Query Queuing on page 343.

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 41 for instructions.

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 214.

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 clear **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 clear **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 StateStoreweb servers: password-based authentication and TLS/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 TLS/SSL Certificate Authentication

1. Create or obtain an TLS/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 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 IODBC driver needed for non-Windows systems, then for the Cloudera ODBC Connector, and finally for the BI tool itself.

```
$ ls -1
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 IODBC driver using its installer
$ open ClouderaImpalaODBC.dmg
Install the Cloudera ODBC Connector using its installer
$ installer_dir=$(pwd)
$ cd /opt/cloudera/impalaodbc
$ ls -1
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 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:**
>
> Prior to CDH 5.7 / Impala 2.5, the Hive JDBC driver did not support connections that use both Kerberos authentication and SSL encryption. If your cluster is running an older release that has this restriction, to use both of these security features with Impala through a JDBC application, use the Cloudera JDBC Connector as the JDBC driver.

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:

- **Complex type considerations:**
  - Queries involving the complex types (ARRAY, STRUCT, and MAP) require notation that might not be available in all levels of JDBC and ODBC drivers. If you have trouble querying such a table due to the driver level or inability to edit the queries used by the application, you can create a view that exposes a “flattened” version of the complex columns and point the application at the view. See Complex Types (CDH 5.5 or higher only) for details.
  - The complex types available in CDH 5.5 / Impala 2.3 and higher are supported by the JDBC `getColumns()` API. Both MAP and ARRAY are reported as the JDBC SQL Type ARRAY, because this is the closest matching Java SQL type. This behavior is consistent with Hive. STRUCT types are reported as the JDBC SQL Type STRUCT.

    To be consistent with Hive's behavior, the TYPE_NAME field is populated with the primitive type name for scalar types, and with the full `toSql()` for complex types. The resulting type names are somewhat inconsistent, because nested types are printed differently than top-level types. For example, the following list shows how `toSQL()` for Impala types are translated to TYPE_NAME values:

```
DECIMAL(10,10)         becomes  DECIMAL
CHAR(10)               becomes  CHAR
```

```
VARCHAR(10)            becomes   VARCHAR
ARRAY<DECIMAL(10,10)>  becomes   ARRAY<DECIMAL(10,10)>
ARRAY<CHAR(10)>        becomes   ARRAY<CHAR(10)>
ARRAY<VARCHAR(10)>     becomes   ARRAY<VARCHAR(10)>
```

Kudu Considerations for DML Statements

Currently, Impala `INSERT`, `UPDATE`, or other DML statements issued through the JDBC interface against a Kudu table do not return JDBC error codes for conditions such as duplicate primary key columns. Therefore, for applications that issue a high volume of DML statements, prefer to use the Kudu Java API directly rather than a JDBC application.

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 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 box 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 box.

   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.

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 Oozie

This section describes tasks for managing Oozie.

Oozie 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 for Oozie High Availability

- 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.

To enable Kerberos authentication, see Enabling Kerberos Authentication Using the Wizard.

For information on setting up TLS/SSL communication with Oozie HA enabled, see Additional Considerations when Configuring TLS/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. Enter the FQDN and port number of the Oozie load balancer. For example:

```
load-bal.example.com:12345
```

6. 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. In addition, Cloudera Manager generates Kerberos credentials for the new Oozie server and regenerates credentials for existing servers.

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.

To enable Kerberos authentication for an Oozie HA-enabled deployment, see Configuring Oozie HA with Kerberos.

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 41 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 is 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.

Dumping and Loading an Oozie Database Using Cloudera Manager

Minimum Required Role: Full Administrator

Oozie is a stateless web application by design. All information about running and completed workflows, coordinators, and bundle jobs are stored in a relational database.

Oozie supports a lightweight embedded (Derby) database, however Cloudera strongly recommends that you use an external database for production systems. For more information, see Supported Databases and Configuring an External Database for Oozie.

The migration tool is not optimized for migrating large databases. If your database size exceeds 1 million rows, Cloudera recommends that you purge it first. See Configuring Oozie Data Purge Settings Using Cloudera Manager on page 239.

This page explains how to dump and load the Oozie database.

Dumping the Oozie Database

To dump your Oozie database:

1. Stop the Oozie server (in HA mode, stop all Oozie servers).
2. In the Cloudera Manager **Admin Console**, go to the Oozie service status page.
3. Select **Actions** > **Stop**. Confirm you want to stop the service by clicking **Stop**.
4. Specify *Database Dump File*:

a. Go to the **Configuration** page.
b. Select **Scope** > **Oozie Server**.
c. Select **Category** > **Database**.
d. Set a file location for the Database Dump File.

5. Select **Actions** > **Dump Database**. Confirm that you want to dump the database to the specified location by clicking **Dump Database**.

During the export process, Cloudera Manager fetches and writes the database content a compressed zip specified by the *Database Dump File* property.

Loading the Oozie Database

To load your Oozie database:

1. Stop the Oozie server (in HA mode, stop all Oozie servers).
2. Install and configure the empty database in which to load your Oozie data. See Configuring an External Database for Oozie. The db.version of the database must match the db.version of the dump file.
3. Select **Actions** > **Create Oozie Database Tables**. Confirm you want to create the database tables by clicking **Create Oozie Database Tables**.
4. Verify *Database Dump File* is set correctly:

   a. In the Cloudera Manager **Admin Console**, click the **Oozie** service.
   b. Go to the **Configuration** page.
   c. Select **Scope** > **Oozie Server**.
   d. Select **Category** > **Database**.
   e. Verify the *Database Dump File* property value.

5. Select **Actions** > **Load Database**. Confirm you want to dump the database to the specified location by clicking **Load Database**.
6. Select **Actions** > **Start**. Confirm you want to start the service by clicking **Start**.

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 14: Oozie Schema - CDH 5 on page 241 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 Using Cloudera Manager on page 13.

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 14: Oozie Schema - CDH 5

	CDH 5.8.0	CDH 5.5.0	CDH 5.4.0	CDH 5.2.0	CDH 5.1.0	CDH 5.0.1
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	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-0.2	email-action-0.1 email-action-0.2	email-action-0.1
hive	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5 hive-action-0.6	hive-action-0.2 hive-action-0.3 hive-action-0.4 hive-action-0.5 hive-action-0.6	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-0.2	hive2-action-0.1 hive2-action-0.2	hive2-action-0.1	hive2-action-0.1		
oozie-bundle	hive2-action-0.1	hive2-action-0.1	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-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-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-sla-0.1 oozie-sla-0.2
oozie-workflow	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 oozie-workflow-0.5	oozie-workflow-0.1 oozie-workflow-0.2 oozie-workflow-0.2.5 oozie-workflow-0.3 oozie-workflow-0.4 oozie-workflow-0.4.5 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	shell-action-0.1 shell-action-0.2 shell-action-0.3
spark	spark-action-0.1 spark-action-0.2	spark-action-0.1	spark-action-0.1			
sqoop	sqoop-action-0.2 sqoop-action-0.3	sqoop-action-0.2 sqoop-action-0.3	sqoop-action-0.2 sqoop-action-0.3	sqoop-action-0.2 sqoop-action-0.3	sqoop-action-0.2 sqoop-action-0.3	sqoop-action-0.2 sqoop-action-0.3

	CDH 5.8.0	CDH 5.5.0	CDH 5.4.0	CDH 5.2.0	CDH 5.1.0	CDH 5.0.1
	sqoop-action-0.4	sqoop-action-0.4	sqoop-action-0.4	sqoop-action-0.4	sqoop-action-0.4	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	ssh-action-0.1 ssh-action-0.2

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.

Enabling Oozie SLA with Cloudera Manager

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

1. In the Cloudera Manager Admin Console, go to the Oozie service.
2. Click the **Configuration** tab.
3. Locate the **Enable SLA Integration** property or search for it by typing its name in the Search box.
4. Select **Enable SLA Integration**. This sets the required values for `oozie.services.ext` and `oozie.service.EventHandlerService.event.listeners` in `oozie-site.xml`.

5. Click **Save Changes** to commit the changes.
6. Restart the Oozie service.

The following properties are set by default when you enable Oozie SLA in Cloudera Manager. You do not have to explicitly define them, unless you want to modify any of these parameters:

```
    oozie.service.SchemaService.wf.schemas
    oozie.service.SchemaService.coord.schemas
    oozie.service.SchemaService.sla.schemas
    oozie.service.ELService.groups
    oozie.service.ELService.constants.wf-sla-submit
    oozie.service.ELService.ext.constants.coord-sla-create
    oozie.service.ELService.functions.coord-sla-create
    oozie.service.ELService.constants.coord-sla-submit
    oozie.service.ELService.functions.coord-sla-submit
    oozie.service.EventHandlerService.filter.app.types
    oozie.service.EventHandlerService.event.queue
    oozie.service.EventHandlerService.queue.size
    oozie.service.EventHandlerService.worker.interval
    oozie.service.EventHandlerService.batch.size
    oozie.service.EventHandlerService.worker.threads
    oozie.sla.service.SLAService.alert.events
    oozie.sla.service.SLAService.capacity
    oozie.sla.service.SLAService.calculator.impl
    oozie.sla.service.SLAService.job.event.latency
    oozie.sla.service.SLAService.check.interval
```

For `oozie.sla.service.SLAService.alert.events`, only `END_MISS` is configured by default. To change the alert events, explicitly set `END_MISS`, `START_MISS`, or `DURATION_MISS`, in **Oozie Server Advanced Configuration Snippet (Safety Valve) for oozie-site.xml**.

For more information on configuring SLA for Oozie, see Oozie SLA Monitoring.

For the complete list of supported configuration properties, see Oozie Configuration Properties.

Setting the Oozie Database Timezone

We recommended 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.JPAService.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 explains 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 15: 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 *n*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.

Configuring Oozie to Enable MapReduce Jobs To Read/Write from Amazon S3

Starting with CDH 5.9, MapReduce jobs controlled by Oozie as part of a workflow can read from and write to Amazon S3. The steps below show you how to enable this capability. Before you begin, you will need your AWS credentials (the appropriate Access key ID and Secret access key obtained from Amazon Web Services for your Amazon S3 bucket). After storing these credentials in the keystore (the JCEKS file), specify the path to this keystore in the Oozie workflow configuration.

> **Note:** This setup is for use in the context of Oozie workflows only, and does not support running shell scripts on Amazon S3 or other types of scenarios.

In the steps below, replace the *path/to/file* with the HDFS directory where the `.jceks` file is located, and replace *access_key_ID* and *secret_access_key* with your AWS credentials.

1. Create the credential store (`.jceks`) and add your AWS access key to it as follows:

```
hadoop credential create fs.s3a.access.key -provider \
jceks://hdfs/path/to/file.jceks -value access_key_id
```

For example:

```
hadoop credential create fs.s3a.access.key -provider \
jceks://hdfs/user/root/awskeyfile.jceks -value AKIAIPVYH....
```

2. Add the AWS secret to this same keystore:

```
hadoop credential create fs.s3a.secret.key -provider \
jceks://hdfs/path/to/file.jceks -value secret_access_key
```

3. Set `hadoop.security.credential.provider.path` to the path of the `.jceks` file in Oozie's `workflow.xml` file in the MapReduce Action's `<configuration>` section so that the MapReduce framework can load the AWS credentials that give access to Amazon S3.

```
<action name="S3job">
    <map-reduce>
        <job-tracker>${jobtracker}</job-tracker>
        <name-node>${namenode}</name-node>
        <configuration>
            <property>
                <name>hadoop.security.credential.provider.path</name>
                <value>jceks://hdfs/path/to/file.jceks</value>
            </property>
            ....
            ....
</action>
```

For more information about Amazon Web Services (AWS) credentials and Amazon S3, see How To Configure Security for Amazon S3.

Managing Solr

You can install the Solr service through the Cloudera Manager installation wizard, using either parcels or packages. See Installing Cloudera 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 41 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 91.

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 Using Cloudera Manager on page 13.

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, 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.cdh4.3.0.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.cdh4.3.0.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.

Using a Load Balancer with Solr

To configure a load balancer:

1. Go to the Solr service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Solr**
4. Select **Category** > **All**
5. Enter the hostname and port number of the load balancer in the **Solr Load Balancer** property in the format `hostname:port number`.

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

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

Migrating Solr Replicas

When you replace a host, migrating replicas on that host to the new host, instead of depending on failure recovery, can help ensure optimal performance.

Where possible, the Solr service routes requests to the proper host. Both `ADDREPLICA` and `DELETEREPLICA` calls can be sent to any host in the cluster.

- For adding replicas, the node parameter ensures the new replica is created on the intended host. If no host is specified, Solr selects a host with relatively fewer replicas.
- For deleting replicas, the request is routed to the host that hosts the replica to be deleted.

Adding replicas can be resource intensive. For best results, add replicas when the system is not under heavy load. For example, do not add additional replicas when heavy indexing is occurring or when MapReduceIndexerTool jobs are running.

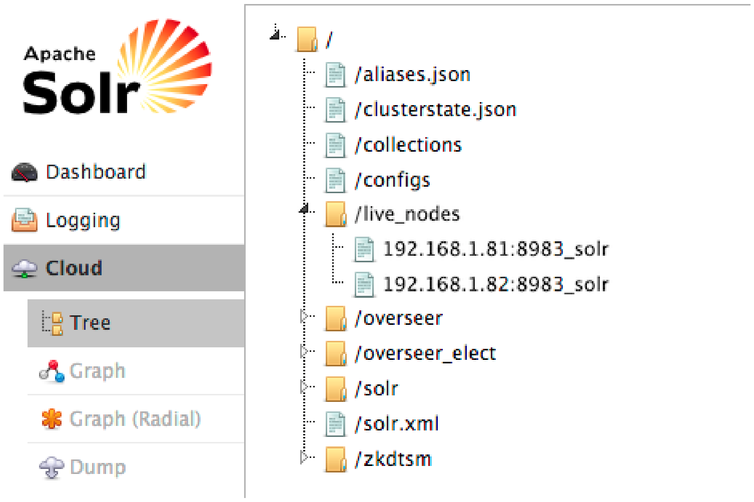
Cloudera recommends using API calls to create and unload cores. Do not use the Cloudera Manager Admin Console or the Solr Admin UI for these tasks.

This procedure uses the following names:

- Host names:
  - `origin` at the IP address 192.168.1.81:8983_solr.
  - `destination` at the IP address 192.168.1.82:8983_solr.
- Collection name `email`
- Replicas:
  - The original replica `email_shard1_replica1`, which is on `origin`.
  - The new replica `email_shard1_replica2`, which will be on `destination`.

To migrate a replica to a new host

1. (Optional) If you want to add a replica to a particular node, review the contents of the `live_nodes` directory on ZooKeeper to find all nodes available to host replicas. Open the Solr Administration User interface, click **Cloud**, click **Tree**, and expand **live_nodes**. The Solr Administration User Interface, including **live_nodes**, might appear as follows:



> **Note:** Information about Solr nodes can also be found in `clusterstate.json`, but that file only lists nodes currently hosting replicas. Nodes running Solr but not currently hosting replicas are not listed in `clusterstate.json`.

2. Add the new replica on `destination` server using the ADDREPLICA API.

```
http://192.168.1.81:8983/solr/admin/collections?action=ADDREPLICA&collection=email&shard=shard1&node=192.168.1.82:8983_solr
```

3. Verify that the replica creation succeeds and moves from recovery state to **ACTIVE**. You can check the replica status in the Cloud view, which can be found at a URL similar to: `http://192.168.1.82:8983/solr/#/~cloud`.

> **Note:** Do not delete the original replica until the new one is in the **ACTIVE** state. When the newly added replica is listed as **ACTIVE**, the index has been fully replicated to the newly added replica. The total time to replicate an index varies according to factors such as network bandwidth and the size of the index. Replication times on the scale of hours are not uncommon and do not necessarily indicate a problem.
>
> You can use the `details` command to get an XML document that contains information about replication progress. Use `curl` or a browser to access a URI similar to:
>
> ```
> http://192.168.1.82:8983/solr/email_shard1_replica2/replication?command=details
> ```
>
> Accessing this URI returns an XML document that contains content about replication progress. A snippet of the XML content might appear as follows:
>
> ```
> ...
> <str name="numFilesDownloaded">126</str>
> <str name="replication StartTime">Tue Jan 21 14:34:43 PST 2014</str>
> <str name="timeElapsed">457s</str>
> <str name="currentFile">4xt_Lucene41_0.pos</str>
> <str name="currentFileSize">975.17 MB</str>
> <str name="currentFileSizeDownloaded">545 MB</str>
> <str name="currentFileSizePercent">55.0</str>
> <str name="bytesDownloaded">8.16 GB</str>
> <str name="totalPercent">73.0</str>
> <str name="timeRemaining">166s</str>
> <str name="downloadSpeed">18.29 MB</str>
> ...
> ```

4. Use the `CLUSTERSTATUS` API to retrieve information about the cluster, including current cluster status:

```
http://192.168.1.81:8983/solr/admin/collections?action=clusterstatus&wt=json&indent=true
```

Review the returned information to find the correct replica to remove. An example of the JSON file might appear as follows:

5. Delete the old replica on `origin` server using the `DELETEREPLICA` API:

```
http://192.168.1.81:8983/solr/admin/collections?action=DELETEREPLICA&collection=email&shard=shard1&replica=core_node2
```

The `DELTEREPLICA` call removes the `datadir`.

Managing Spark

Apache Spark is a general framework for distributed computing that offers high performance for both batch and interactive processing.

To run applications distributed across a cluster, Spark requires a cluster manager. Cloudera supports two cluster managers: YARN and Spark Standalone. When run on YARN, Spark application processes are managed by the YARN ResourceManager and NodeManager roles. When run on Spark Standalone, Spark application processes are managed by Spark Master and Worker 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.

Related Information

- Spark Guide
- Monitoring Spark Applications
- Tuning Spark Applications on page 304
- Spark Authentication
- Cloudera Spark forum
- Apache Spark documentation

This section describes how to manage Spark services.

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 41.

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 Application Overview.

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

> **Important:** This item is deprecated and will be removed in a future release. Cloudera supports items that are deprecated until they are removed. For more information about deprecated and removed items, see Deprecated Items.

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 253.

For information on Spark applications, see Spark Application Overview.

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

1. 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=hdfs://namenode_host:namenode_port/user/spark/applicationHistory
spark.eventLog.enabled=true
```

   or

```
spark.eventLog.dir=hdfs://name_service_id/user/spark/applicationHistory
spark.eventLog.enabled=true
```

   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. This causes Spark applications to write their history to the directory that the History Server reads.

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 Installing the 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 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 box 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 box.

   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 can install the Sqoop 2 service as part of the CDH installation.

Note: Sqoop 2 is 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 choose to add it in Custom Services. 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 41 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 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 YARN (MRv2) and MapReduce (MRv1)

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 also run in a YARN cluster).

The MapReduce (MRv2) or YARN architecture splits the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager and per-application ApplicationMasters. With MRv2, the ResourceManager and per-host NodeManagers 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 run 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 396.
- For information on configuring MapReduce and YARN resource management features, see Resource Management on page 320.

Defaults and Recommendations

- 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.
- 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.
- 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 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 260 to easily migrate MapReduce configurations to YARN. The wizard performs all the steps (Switching Between MapReduce and YARN Services on page 257, Updating Services Dependent on MapReduce on page 257, and Configuring Alternatives Priority for Services Dependent on MapReduce on page 257) 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 **Home** > **Status** tab, 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               |
+--------------------+
```

   b. Back up the database:

```
$ mysqldump -uroot -pcloudera amon > /safe_backup_directory/amon.sql
```

   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 host. 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 host.

For detailed information on migrating from MapReduce to YARN, see [Migrating from MapReduce (MRv1) to MapReduce (MRv2)](#).

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 Services Dependent on MapReduce

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 for Services Dependent on MapReduce

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.

Configuring MapReduce To Read/Write With Amazon Web Services

These are the steps required to configure MapReduce to read and write with AWS.

1. Save your AWS access key in a `.jceks` file in HDFS.

```
hadoop credential create fs.s3a.access.key -provider \
    jceks://hdfs/<hdfs directory>/<file name>.jceks -value <AWS access key id>
```

2. Put the AWS secret in the same `.jceks` file created in previous step.

```
hadoop credential create fs.s3a.secret.key -provider \
    jceks://hdfs/<hdfs directory>/<file name>.jceks -value <AWS secret access key>
```

3. Set your `hadoop.security.credential.provider.path` to the path of the `.jceks` file in the job configuration so that the MapReduce framework loads AWS credentials from the `.jceks` file in HDFS. The following example shows a Teragen MapReduce job that writes to an S3 bucket.

```
hadoop jar <path to the Hadoop MapReduce example jar file> teragen \
     -Dhadoop.security.credential.provider.path= \
     jceks://hdfs/<hdfs directory>/<file name>.jceks \
     100 s3a://<bucket name>/teragen1
```

You can specify the variables *<hdfs directory>, <file name>, <AWS access key id>*, and *<AWS secret access key>*. *<hdfs directory>* is the HDFS directory where you store the `.jceks` file. *<file name>* is the name of the `.jceks` file in HDFS.

To configure Oozie to submit S3 MapReduce jobs, see Configuring Oozie to Enable MapReduce Jobs To Read/Write from Amazon S3 on page 246.

Managing YARN

For an overview of computation frameworks, insight into their usage and restrictions, and examples of common tasks they perform, see Managing YARN (MRv2) and MapReduce (MRv1) on page 255.

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. Select **YARN (MR2 Included)** and click **Continue**.
3. Select 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 box 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 box.

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 YARN (MRv2) and MapReduce (MRv1) on page 255 for more configuration specifics; and, for detailed tuning advice with sample configurations, see Tuning YARN on page 310.

Table 16: 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 host
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 Cloudera Manager logo to return to the **Home** page.
   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 Fair Scheduler. 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 YARN (MRv2) and MapReduce (MRv1) Schedulers on page 337.

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 Using Cloudera Manager]() on page 13.

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 Using Cloudera Manager]() on page 13.

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 326.

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 `hdfs` 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 Spark on YARN for Long-Running Applications]().

Task Process Exit Codes

All YARN tasks on the NodeManager are run in a JVM. When a task runs 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.

The task JVM might exit with a non-zero code for multiple reasons, though there is no exhaustive list. Exit codes can be split into two categories:

- Set by the JVM based on the OS signal received by the JVM
- 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 preemption (if that is configured) or a speculative run. 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 running 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 YARN ACLs

An Access Control List (ACL) is a list of specific permissions or controls that allow individual users or groups to perform specific actions upon specific objects, as well as defining what operations are allowed on a given object. YARN ACLs do not deny access; rather, they identify a user, list of users, group, or list of groups who can access a particular object.

Like HDFS ACLs, YARN ACLs provide a way to set different permissions for specific named users or named groups. ACLs enhance the traditional permissions model by defining access control for arbitrary combinations of users and groups instead of a single owner/user or a single group.

YARN ACL Rules and Syntax

This section describes the rules governing YARN ACLs and includes syntax examples.

YARN ACL Rules

All YARN ACLs must adhere to the following rules:

- **Special Values:**
  - The wildcard character (*) indicates that everyone has access.

    > **Note:** You cannot use the wildcard (*) character along with a list of users and/or groups in the same ACL. If you use the wildcard character it must be the *only* item in the ACL.

  - A single space entry indicates that no one has access.
- If there are no spaces in an ACL, then all entries (the listed users and/or groups) are considered authorized users.
- Group names in YARN Resource Manager ACLs are case sensitive. So, if you specify an uppercase group name in the ACL, it will not match the group name resolved from the Active Directory because Active Directory group names are resolved in lowercase.
- If an ACL starts with a single space, then it must consist of groups only.
- All entries after the occurrence of a second single space in an ACL are ignored.
- There are no ACLs that deny access to a user or group. However, if you wish to block access to an operation entirely, enter a value for a non-existent user or group (for example,`'NOUSERS NOGROUPS'`), or simply enter a single space. By doing so, you ensure that no user or group maps to a particular operation by default.
- If you wish to deny only a certain set of users and/or groups, specify every single user and/or group that requires access. Users and/or groups that are not included are "implicitly" denied access.

YARN ACL Syntax

Following are examples of YARN ACL syntax:

> **Note:** In all cases where a single space is required, you will see: `<single space>`.

- Users only

```
user1,user2,userN
```

Use a comma-separated list of user names. Do not place spaces after the commas separating the users in the list.

- Groups only

```
<single space>HR,marketing,support
```

You *must* begin group-only ACLs with a single space. Group-only ACLs use the same syntax as users, except each entry is a group name rather than user name.

- Users and Groups

```
fred,alice,haley<single space>datascience,marketing,support
```

A comma-separated list of user names, followed by a single space, followed by a comma-separated list of group names. This sample ACL authorizes access to users “fred”, “alice”, and “haley”, and to those users in the groups “datascience”, “marketing”, and “support”.

Examples

The following ACL entry authorizes access only to the members of “my_group”:

```
<single space>my_group
```

The following ACL entry authorizes access to anyone:

```
*
```

The following ACL authorizes access to the users “john”, “jane”, and the group “HR”:

```
john,jane<single space>HR
```

In this example, six groups (“group_1” through “group_6”) are defined in the system. The following ACL authorizes access to a subset of the defined groups, allowing access to all members of groups 1 through 5 (and implicitly denies access to members of the group “group_6”):

```
<single space>group_1,group_2,group_3,group_4,group_5
```

Activating YARN ACLs

> **Important:** See YARN Admin ACL on page 264 before activating YARN ACLs, because you must configure the YARN Admin ACL first, before activation.

In a default Cloudera Manager managed YARN deployment, ACL checks are turned on but do not provide any security, which means that any user can execute administrative commands or submit an application to any YARN queue. To provide security the ACL must be changed from its default value, the wildcard character (*).

In non-Cloudera Manager managed clusters, the default YARN ACL setting is `false`, and ACLs are turned off and provide security out-of-the-box.

Activate YARN ACLs via the `yarn.acl.enable` property (values are either `true` or `false`):

```
<property>
   <name>yarn.acl.enable</name>
   <value>true</value>
</property>
```

YARN ACLs are independent of HDFS or protocol ACLs, which secure communications between clients and servers at a low level.

YARN ACL Types

This section describes the types of YARN ACLs available for use:

- YARN Admin ACL on page 264

  (`yarn.admin.acl`)
- Queue ACL on page 264

  (`aclSubmitApps and aclAdministerApps`)
- Application ACL

  (`mapreduce.job.acl-view-job` and `mapreduce.job.acl-modify-job`)

YARN Admin ACL

Use the YARN Admin ACL to allow users to run YARN administrator sub-commands, which are executed via the `yarn rmadmin <command>`.

> **Important:** The YARN Admin ACL is triggered and applied *only* when you run YARN sub-commands via `yarn rmadmin <cmd>`. If you run other YARN commands via the YARN command line (for example, starting the Resource or Node Manager), it does not trigger the YARN Admin ACL check or provide the same level of security.

The default YARN Admin ACL is set to the wildcard character (*), meaning all users and groups have YARN Administrator access and privileges. So after YARN ACL enforcement is enabled, (via the `yarn.acl.enable` property) every user has YARN ACL Administrator access. Unless you wish for all users to have YARN Admin ACL access, edit the `yarn.admin.acl` setting upon initial YARN configuration, and before enabling YARN ACLs.

A typical YARN Admin ACL looks like the following, where the system's Hadoop administrator and multiple groups are granted access:

```
hadoopadmin<space>yarnadmgroup,hadoopadmgroup
```

Queue ACL

Use Queue ACLs to identify and control which users and/or groups can take actions on particular queues. Configure Queue ACLs using the aclSubmitApps and aclAdministerApps properties, which are set per queue. Queue ACLs are scheduler dependent, and the implementation and enforcement differ per scheduler type.

> **Note:** Cloudera only supports the Fair Scheduler in CDH. Cloudera does not support Scheduler Reservations (including `aclAdministerReservations`, `aclListReservations`, and `aclSubmitReservations`) and their related ACLs.

Unlike the YARN Admin ACL, Queue ACLs are not enabled and enforced by default. Instead, you must explicitly enable Queue ACLs. Queue ACLs are defined, per queue, in the Fair Scheduler configuration. By default, neither of the Queue ACL property types is set on any queue, and access is allowed or open to any user.

The users and groups defined in the `yarn.admin.acl` are considered to be part of the Queue ACL, `aclAdministerApps`. So any user or group that is defined in the `yarn.admin.acl` can submit to any queue and kill any running application in the system.

The aclSubmitApps Property

Use the Queue ACL `aclSubmitApps` property type to enable users and groups to submit or add an application to the queue upon which the property is set. To move an application from one queue to another queue, you must have Submit permissions for both the queue in which the application is running, and the queue into which you are moving the application. You must be an administrator to set Admin ACLs; contact your system administrator to request Submit permission on this queue.

The aclAdministerApps Property

Use the Queue ACL `aclAdministerApps` property type to enable all actions defined in the aclSubmitApps property, plus any administrative actions that have been defined (the only administrative action currently defined and supported in this context is killing an application).

> **Important:** The users and groups defined in the `yarn.admin.acl` are considered to be part of the Queue ACL, `aclAdministerApps`. So any user or group that is defined in the `yarn.admin.acl` can submit to any queue and kill any running application in the system.

Following is an example of a Queue ACL with both types defined. Note that the single space in `aclAdministerApps` indicates a group-only rule:

```
<queue name="Marketing">
  <aclSubmitApps>john,jane</aclSubmitApps>
  <aclAdministerApps><single space>others</aclAdministerApps>
</queue>
```

Queue ACL Evaluation

The better you understand how Queue ACLs are evaluated, the more prepared you are to define and configure them. First, you should have a basic understanding of how Fair Scheduler queues work.

CDH Fair Scheduler supports hierarchical queues, all of which descend from a root queue, which is automatically created and defined within the system when the Scheduler starts.

Available resources are distributed among the children (“leaf” queues) of the root queue in a typical fair scheduling fashion. Then, the children distribute their assigned resources to their children in the same fashion.

As mentioned earlier, applications are scheduled on leaf queues only. You specify queues as children of other queues by placing them as sub-elements of their parents in the Fair Scheduler allocation file (`fair-scheduler.xml`). The default Queue ACL setting for all parent and leaf queues is “ “ (a single space), which means that by default, no one can access any of these queues.

Queue ACL inheritance is enforced by assessing the ACLs defined in the queue hierarchy in a bottom-up order to the root queue. So within this hierarchy, access evaluations start at the level of the bottom-most leaf queue. If the ACL does not provide access, then the parent Queue ACL is checked. These evaluations continue upward until the root queue is checked.

Queue ACLs do not interact directly with the placement policy rules (the rules that determine the pools to which applications and queries are assigned) and are not part of the placement policy rules, which are executed before the ACLs are checked. The policy rules return a final result in the form of a queue name. The queue is then evaluated for access, as described earlier. The Queue ACL allows or denies access to this final queue, which means that an application can be rejected even if the placement policy returns back a queue.

> **Important:**
>
> In all YARN systems, the default setting for the root queue is reversed compared to all other queues–the root queue has a default setting of “*”, which means everyone has access:
>
> | Queue ACL Property Type | Default Values | |
> |---|---|---|
> | | Root Queue | All Other Queues |
> | aclSubmitApps | * | “ “ (single space) |
> | aclAdministerApps | * | “ “ (single space) |
>
> So even when the Queue ACLs are turned on by default, everyone has access because the root queue ACL is inherited by all the leaf queues.

Best practice: A best practice for securing an environment is to set the root queue aclSubmitApps ACL to <single space>, and specify a limited set of users and groups in aclAdministerApps. Set the ACLs for all other queues to provide submit or administrative access as appropriate.

The order in which the two types of Queue ACLs are evaluated is always:

1. aclSubmitApps
2. aclAdministerApps

The following diagram shows the evaluation flow for Queue ACLs:

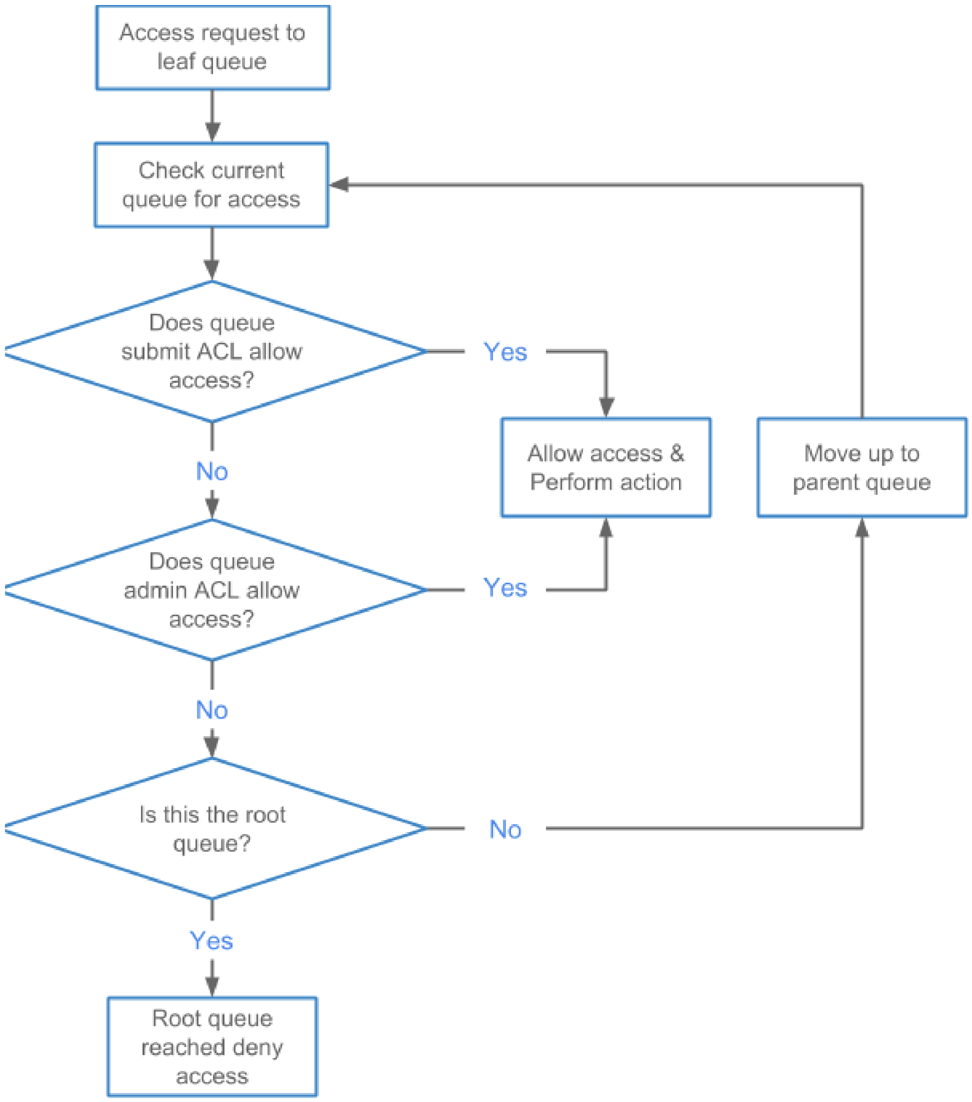


Figure 4: Queue ACL Evaluation Flow

Application ACLs

Use Application ACLs to provide a user and/or group–neither of whom is the owner–access to an application. The most common use case for Application ACLs occurs when you have a team of users collaborating on or managing a set of applications, and you need to provide read access to logs and job statistics, or access to allow for the modification of a job (killing the job) and/or application. Application ACLs are set per application and are managed by the application owner.

Users who start an application (the owners) always have access to the application they start, which includes the application logs, job statistics, and ACLs. No other user can remove or change owner access. By default, no other users have access to the application data because the Application ACL defaults to " " (single space), which means no one has access.

MapReduce

Create and use the following MapReduce Application ACLs to view YARN logs:

- `mapreduce.job.acl-view-job`

  Provides read access to the MapReduce history and the YARN logs.

- `mapreduce.job.acl-modify-job`

Provides the same access as `mapreduce.job.acl-view-job`, and also allows the user to modify a running job.

> **Note:** Job modification is currently limited to killing the job. No other YARN system modifications are supported.

During a search or other activities, you may come across the following two legacy settings from MapReduce; they are not supported by YARN. Do *not* use them:

- `mapreduce.cluster.acls.enabled`
- `mapreduce.cluster.administrators`

Spark

Spark ACLs follow a slightly different format, using a separate property for users and groups. Both user and group lists use a comma-separated list of entries. The wildcard character "*" allows access to anyone, and the single space " " allows access to no one. Enable Spark ACLs using the property `spark.acls.enable`, which is set to `false` by default (not enabled) and must be changed to `true` to enforce ACLs at the Spark level.

Create and use the following Application ACLs for the Spark application:

- Set `spark.acls.enable` to `true` (default is `false`).
- Set `spark.admin.acls` and `spark.admin.acls.groups` for administrative access to all Spark applications.
- Set `spark.ui.view.acls` and `spark.ui.view.acls.groups` for view access to the specific Spark application.
- Set `spark.modify.acls` and `spark.modify.acls.groups` for administrative access to the specific Spark application.

Refer to Spark Security and Spark Configuration Security for additional details.

Viewing Application Logs

The MapReduce Application ACL `mapreduce.job.acl-view-job` determines whether or not you can view an application log, and access is evaluated via the following ACLs:

- YARN Admin and Queue ACLs
- Application ACLs

After an application is in the "finished" state, logs are aggregated, depending on your cluster setup. You can access the aggregated logs via the MapReduce History server web interface. Aggregated logs are stored on shared cluster storage, which in most cases is HDFS. You can also share log aggregation via storage options like S3 or Azure by modifying the `yarn.nodemanager.remote-app-log-dir` setting in Cloudera Manager to point to either S3 or Azure, which should already be configured.

The shared storage on which the logs are aggregated helps to prevent access to the log files via file level permissions. Permissions on the log files are also set at the file system level, and are enforced by the file system: the file system can block any user from accessing the file, which means that the user cannot open/read the file to check the ACLs that are contained within.

In the cluster storage use case of HDFS, you can only access logs that are aggregated via the:

- Application owner
- Group defined for the MapReduce History server

When an application runs, generates logs, and then places the logs into HDFS, a path/structure is generated (for example: `/tmp/logs/john/logs/application_1536220066338_0001`). So access for the application owner "john" might be set to `700`, which means `read`, `write`, `execute`; no one else can view files underneath this directory. If you don't have HDFS access, you will be denied access. Command line users identified in `mapreduce.job.acl-view-job` are also denied access at the file level. In such a use case, the Application ACLs stored inside the aggregated logs will never be evaluated because the Application ACLs do not have file access.

For clusters that do not have log aggregation, logs for running applications are kept on the node where the container runs. You can access these logs via the Resource Manager and Node Manager web interface, which performs the ACL checks.

Killing an Application

The Application ACL `mapreduce.job.acl-modify-job` determines whether or not a user can modify a job, but in the context of YARN, this only allows the user to kill an application. The kill action is application agnostic and part of the YARN framework. Other application types, like MapReduce or Spark, implement their own kill action independent of the YARN framework. MapReduce provides the kill actions via the `mapred` command.

For YARN, the following three groups of users are allowed to kill a running application:

- The application owner
- A cluster administrator defined in `yarn.admin.acl`
- A queue administrator defined in `aclAdministerApps` for the queue in which the application is running

Note that for the queue administrators, ACL inheritance applies, as described earlier.

Application ACL Evaluation

The better you understand how YARN ACLs are evaluated, the more prepared you will be to define and configure the various YARN ACLs available to you. For example, if you enable user access in Administrator ACLs, then you must be aware that user may have access to/see sensitive data, and should plan accordingly. So if you are the administrator for an entire cluster, you also have access to the logs for running applications, which means you can view sensitive information in those logs associated with running the application.

Best Practice: A best practice for securing an environment is to set the YARN Admin ACL to include a limited set of users and or groups.

The following diagram shows the evaluation flow for Application ACLs:

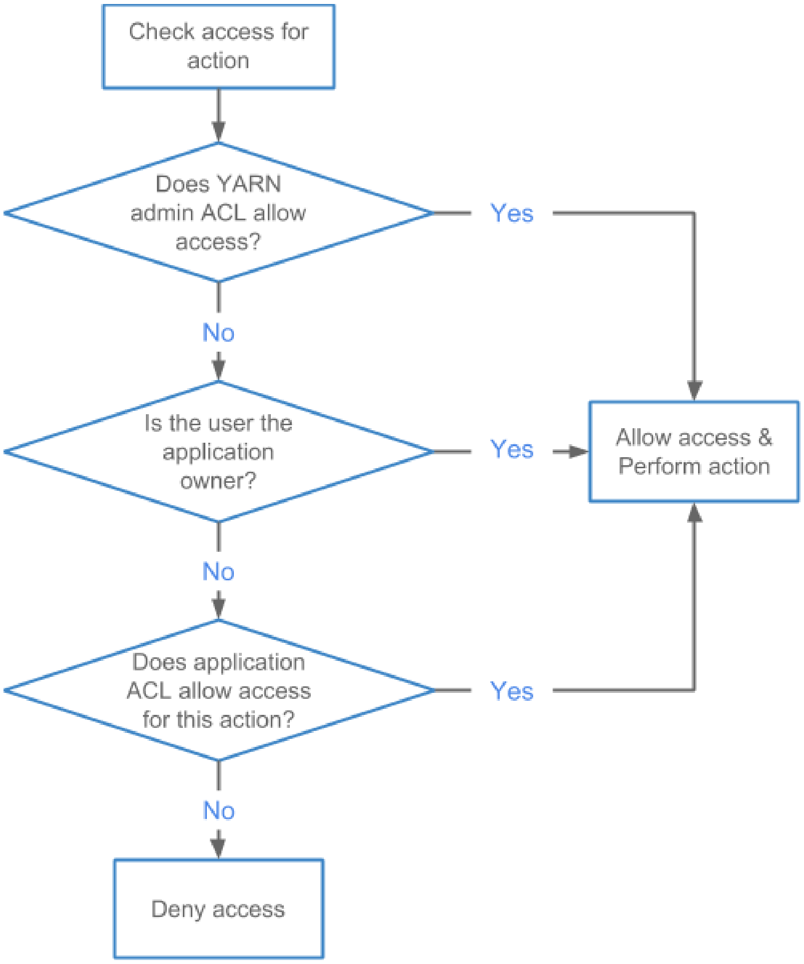


Figure 5: Application ACL Evaluation Flow

The following diagram shows a sample queue structure, starting with leaf queues on the bottom, up to root queue at the top; use it to follow the examples of killing an application and viewing a log:

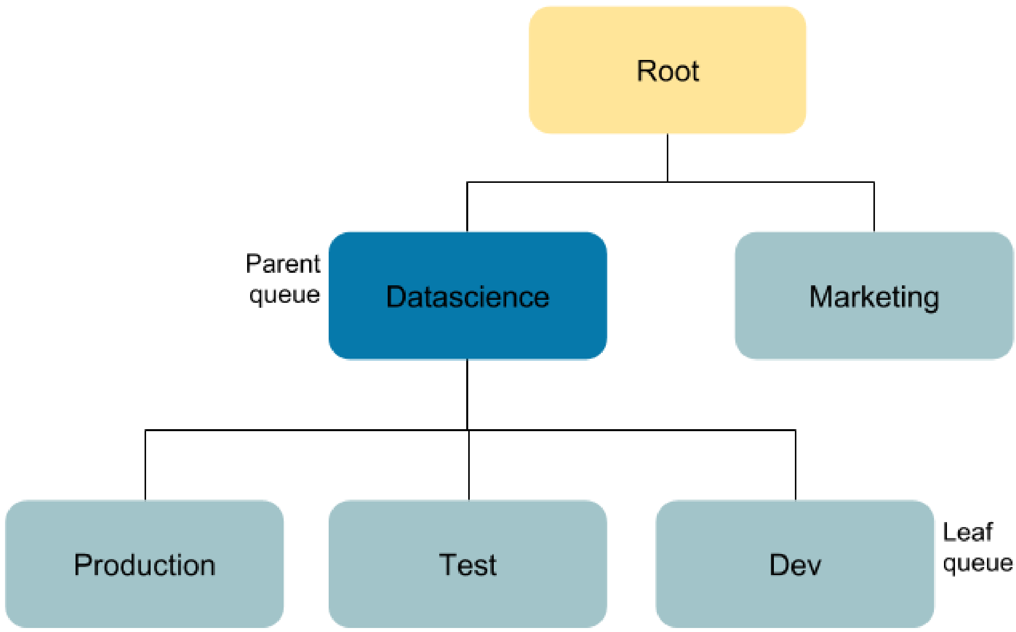


Figure 6: Queue Structure

Example: Killing an Application in the Queue "Production"

For this Application ACL evaluation flow example, assume the following for `application_1536220066338_0001` running in the queue "Production":

- Application owner: John
- "Datascience" queue administrator: Jane
- YARN cluster administrator: Bob

In this use case, John attempts to kill the application (see Killing an Application on page 269), which is allowed because he is the application owner.

Working as the queue administrator, Jane attempts to kill a job in the queue "Production", which she can do as the queue administrator of the parent queue.

Bob is the YARN cluster administrator and he is also listed as a user in the Admin ACL. He attempts to kill the job for which he is not the owner, but because he is the YARN cluster administrator, he can kill the job.

Example: Moving the Application and Viewing the Log in the Queue "Test"

For this Application ACL evaluation flow example, assume the following for `application_1536220066338_0002` running in the queue "Test":

- Application owner: John
- "Marketing" and "Dev" queue administrator: Jane
- Jane has log view rights via the `mapreduce.job.acl-view-job` ACL
- YARN cluster administrator: Bob

In this use case, John attempts to view the logs for his job, which is allowed because he is the application owner.

Jane attempts to access `application_1536220066338_0002` in the queue "Test" to move the application to the "Marketing" queue. She is denied access to the "Test" queue via the queue ACLs–so she cannot submit to or administer the queue "Test". She is also unable to kill a job running in queue "Test". She then attempts to access the logs for `application_1536220066338_0002` and is allowed access via the `mapreduce.job.acl-view-job` ACL.

Bob attempts to access `application_1536220066338_0002` in the queue "Test" to move the application to the "Marketing" queue. As the YARN cluster administrator, he has access to all queues and can move the application.

> **Note:** Permissions on the log files are also set at the filesystem level and are enforced by the filesystem: the filesystem can block you from accessing the file, which means that you can not open/read the file to check the ACLs that are contained in the file.

Configuring and Enabling YARN ACLs

To configure YARN ACLs, refer to Configuring ACLs on page 331.

To enable YARN ACLs, refer to Enabling ACLs on page 331.

Managing MapReduce

For an overview of computation frameworks, insight into their usage and restrictions, and examples of common tasks they perform, see Managing YARN (MRv2) and MapReduce (MRv1) on page 255.

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 YARN (MRv2) and MapReduce (MRv1) Schedulers on page 337.

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 Using Cloudera Manager on page 13.

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 Using Cloudera Manager on page 13.

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 Using Cloudera Manager on page 13.
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 ZooKeeper

This topic describes how to add, remove, and replace ZooKeeper roles.

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.

Adding a ZooKeeper Service Using Cloudera Manager

Minimum Required Role: Full Administrator

When adding the ZooKeeper service, the **Add Service** wizard automatically initializes the data directories.

When you add Zookeeper servers to an existing ensemble, a rolling restart of all zookeeper is required in order to allow all zookeeper servers to have the same configurations

If you quit the **Add Service** wizard or it does not finish successfully, you can initialize the directories outside the wizard by following 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 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.

Replacing a Zookeeper Disk Using Cloudera Manager

Minimum Required Role: Full Administrator

1. In Cloudera Manager, update the **Data Directory** and **Transaction Log Directory** settings.
2. Stop a single ZooKeeper role.
3. Move the contents to the new disk location (modify mounts as needed). Make sure the permissions and ownership are correct.
4. Start the ZooKeeper role.
5. Repeat steps 2-4 for any remaining ZooKeeper roles.

Replacing a ZooKeeper Role Using Cloudera Manager

Minimum Required Role: Full Administrator

1. Go to **ZooKeeper Instances**.
2. Stop the ZooKeeper role on the old host.
3. Confirm the ZooKeeper service has elected one of the remaining hosts as a leader on the ZooKeeper Status page. See Confirming the Election Status of a ZooKeeper Service on page 275.
4. On the **ZooKeeper Instances** page, remove the ZooKeeper role from old host.
5. Add a new ZooKeeper role on the new host.
6. Start the new ZooKeeper role.
7. Confirm on the **ZooKeeper Status** page that there is a leader and all other hosts are followers.
8. Restart the other ZooKeeper roles.
9. Restart any dependent services such as HBase, HDFS Failover Controllers with HDFS High Availability, YARN or Mapreduce v1 with High Availability, or other service.

10. Perform a failover to make one HDFS NameNode active. See Manually Failing Over to the Standby NameNode Using Cloudera Manager on page 391.

Adding or Deleting a ZooKeeper Role on an Unmanaged Cluster

Minimum Required Role: Full Administrator

For information on administering ZooKeeper from the command line, see the ZooKeeper Getting Started Guide.

Replacing a ZooKeeper Role on an Unmanaged Cluster

Minimum Required Role: Full Administrator

These instructions assume you are using ZooKeeper from the command line. For more information, see the ZooKeeper Getting Started Guide.

1. Stop the ZooKeeper role on the old host.
2. Confirm the ZooKeeper Quorum has elected a leader. See Confirming the Election Status of a ZooKeeper Service on page 275.
3. Add a new ZooKeeper role on the new server.
4. Identify the `dataDir` location from the `zoo.cfg` file. This defaults to `/var/lib/zookeeper`.
5. Identify the ID number for the ZooKeeper Server from the `myid` file in the configuration: `cat /var/lib/zookeeper/myid`
6. On all the ZooKeeper hosts, edit the `zoo.cfg` file so the server ID references the new server hostname. For example:

```
server.1=zk1.example.org:3181:4181
server.2=zk2.example.org:3181:4181
server.4=zk4.example.org:3181:4181
```

7. Restart the ZooKeeper hosts.
8. Confirm the ZooKeeper Quorum has elected a leader and the other hosts are followers. See Confirming the Election Status of a ZooKeeper Service on page 275.
9. Restart any dependent services such as HBase, HDFS Failover Controllers with HDFS High Availability, or YARN or Mapreduce v1 with High Availability.
10. Perform a failover to make one HDFS NameNode active. See Manually Failing Over to the Standby NameNode Using the Command Line on page 392.

Confirming the Election Status of a ZooKeeper Service

Determining the election status of a ZooKeeper host requires that you have installed telnet or nc (netcat), running from a host with network access to the ZooKeeper host. The default ZooKeeper client port is 2181. Run the following command against each ZooKeeper host:

```
echo "stat" | nc server.example.org 2181 | grep Mode
```

For example, a follower host would return the message:

```
Mode: follower
```

You can use telnet, if you prefer.

```
$ telnet server.example.org 2181
```

Sample output would be similar to the following.

```
Trying 10.1.2.154...
Connected to server.example.org.
Escape character is '^]'.
stat
Zookeeper version: 3.4.5-cdh5.4.4--1, built on 07/06/2015 23:54 GMT
...
```

```
Latency min/avg/max: 0/1/40
Received: 631
Sent: 677
Connections: 7
Outstanding: 0
Zxid: 0x30000011a
Mode: follower            <----
Node count: 40
Connection closed by foreign host.
```

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 1

1. Add the following entries to the Sqoop 1 Client Client Advanced Configuration Snippet (Safety Valve)
    - `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. Re-deploy the client configuration.

Sqoop 2

1. Add the following entries to the **Sqoop 2 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.

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:** Work with your network administrators and hardware vendors to ensure 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 do its best to share useful experiences.

Disable the tuned Service

If your cluster hosts are running RHEL/CentOS 7.x, disable the "tuned" service by running the following commands:

1. Ensure that the tuned service is started:

```
systemctl start tuned
```

2. Turn the tuned service off:

```
tuned-adm off
```

3. Ensure that there are no active profiles:

```
tuned-adm list
```

   The output should contain the following line:

```
No current active profile
```

4. Shutdown and disable the tuned service:

```
systemctl stop tuned
systemctl disable tuned
```

Disabling Transparent Hugepages (THP)

Most Linux platforms supported by CDH 5 include a feature called ***transparent hugepages***, 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.

To see whether transparent hugepages are enabled, run the following commands and check the output:

```
$ cat defrag_file_pathname
$ cat enabled_file_pathname
```

- `[always] never` means that transparent hugepages is enabled.
- `always [never]` means that transparent hugepages is disabled.

To disable Transparent Hugepages, perform the following steps on all cluster hosts:

1. (**Required for hosts running RHEL/CentOS 7.x.**) To disable transparent hugepages on reboot, add the following commands to the `/etc/rc.d/rc.local` file on all cluster hosts:
   - **RHEL/CentOS 7.x:**

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
echo never > /sys/kernel/mm/transparent_hugepage/defrag
```

   - **RHEL/CentOS 6.x**

```
echo never > /sys/kernel/mm/redhat_transparent_hugepage/defrag
echo never > /sys/kernel/mm/redhat_transparent_hugepage/enabled
```

   - **Ubuntu/Debian, OL, SLES**:

```
echo never > /sys/kernel/mm/transparent_hugepage/defrag
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

   Modify the permissions of the `rc.local` file:

```
chmod +x /etc/rc.d/rc.local
```

2. If your cluster hosts are running RHEL/CentOS 7.x, modify the GRUB configuration to disable THP:
   - Add the following line to the `GRUB_CMDLINE_LINUX` options in the `/etc/default/grub` file:

```
transparent_hugepage=never
```

   - Run the following command:

```
grub2-mkconfig -o /boot/grub2/grub.cfg
```

3. Disable the tuned service, as described above.

You can also disable transparent hugepages interactively (but remember this will not survive a reboot).

To disable transparent hugepages temporarily as root:

```
# echo 'never' > defrag_file_pathname
# echo 'never' > enabled_file_pathname
```

To disable transparent hugepages temporarily using sudo:

```
$ sudo sh -c "echo 'never' > defrag_file_pathname"
$ sudo sh -c "echo 'never' > enabled_file_pathname"
```

Setting the vm.swappiness Linux Kernel Parameter

The Linux kernel parameter, `vm.swappiness`, is a value from 0-100 that controls the swapping of application data (as anonymous pages) from physical memory to virtual memory on disk. The higher the value, the more aggressively inactive processes are swapped out from physical memory. The lower the value, the less they are swapped, forcing filesystem buffers to be emptied.

On most systems, `vm.swappiness` is set to 60 by default. This is not suitable for Hadoop clusters because processes are sometimes swapped even when enough memory is available. This can cause lengthy garbage collection pauses for important system daemons, affecting stability and performance.

Cloudera recommends that you set `vm.swappiness` to a value between 1 and 10, **preferably 1**, for minimum swapping on systems where the RHEL kernel is 2.6.32-642.el6 or higher.

To view your current setting for `vm.swappiness`, run:

```
cat /proc/sys/vm/swappiness
```

To set `vm.swappiness` to 1, run:

```
sudo sysctl -w vm.swappiness=1
```

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.

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 `mapreduce.shuffle.manage.os.cache`, to `true` (default). To further tune performance, adjust the value of `mapreduce.shuffle.readahead.bytes`. The default value is 4 MB.
- To enable this feature for MapReduce, set the `mapred.tasktracker.shuffle.fadvise` to `true` (default). To further tune performance, adjust the value of `mapred.tasktracker.shuffle.readahead.bytes`. The default value is 4 MB.

IFile Reader

Enabling IFile readahead increases the performance of merge operations. To enable this feature for either MRv1 or YARN, set `mapreduce.ifile.readahead` to `true` (default). To further tune the performance, adjust the value of `mapreduce.ifile.readahead.bytes`. 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 `mapreduce.tasktracker.outofband.heartbeat` to `true` for TaskTracker to 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`*. This copies the local JAR files to HDFS and uses the distributed cache mechanism to ensure they are available on the task nodes and added to the task classpath.

The advantage of this, over `JobConf.setJar`, is that if the JAR is on a task node, it does not 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 does not, 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 and Configuring Data Compression

For an overview of compression, see Data Compression.

Guidelines for Choosing a Compression Type

- 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 and Spark, 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. 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.
  - **MRv2**

```
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
```

  - **MRv1**

```
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

Configuring Data Compression Using Cloudera Manager

To configure support for LZO using Cloudera Manager, you must install the GPL Extras parcel, then configure services to use it. See Installing the GPL Extras Parcel and Configuring Services to Use the GPL Extras Parcel on page 276.

Configuring Data Compression Using the Command Line

Important:

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

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 with MapReduce. Use the same method for LZO, with the codec `com.hadoop.compression.lzo.LzopCodec` instead.

Tuning Hive

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

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

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

Heap Size and Garbage Collection for Hive Components

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

Memory Recommendations

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

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

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

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

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

Configuring Heap Size and Garbage Collection

Using Cloudera Manager

To configure heap size and garbage collection for HiveServer2:

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

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

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

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

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

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

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

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

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

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

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

Using the Command Line

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

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

The following example shows a configuration with the following settings:

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

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

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

```
  fi
fi

export HADOOP_HEAPSIZE=2048
```

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

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

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

HiveServer2 Performance Tuning and Troubleshooting

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

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

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

Symptoms Displayed When HiveServer2 Heap Memory is Full

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

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

Troubleshooting

HiveServer2 Service Crashes

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

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

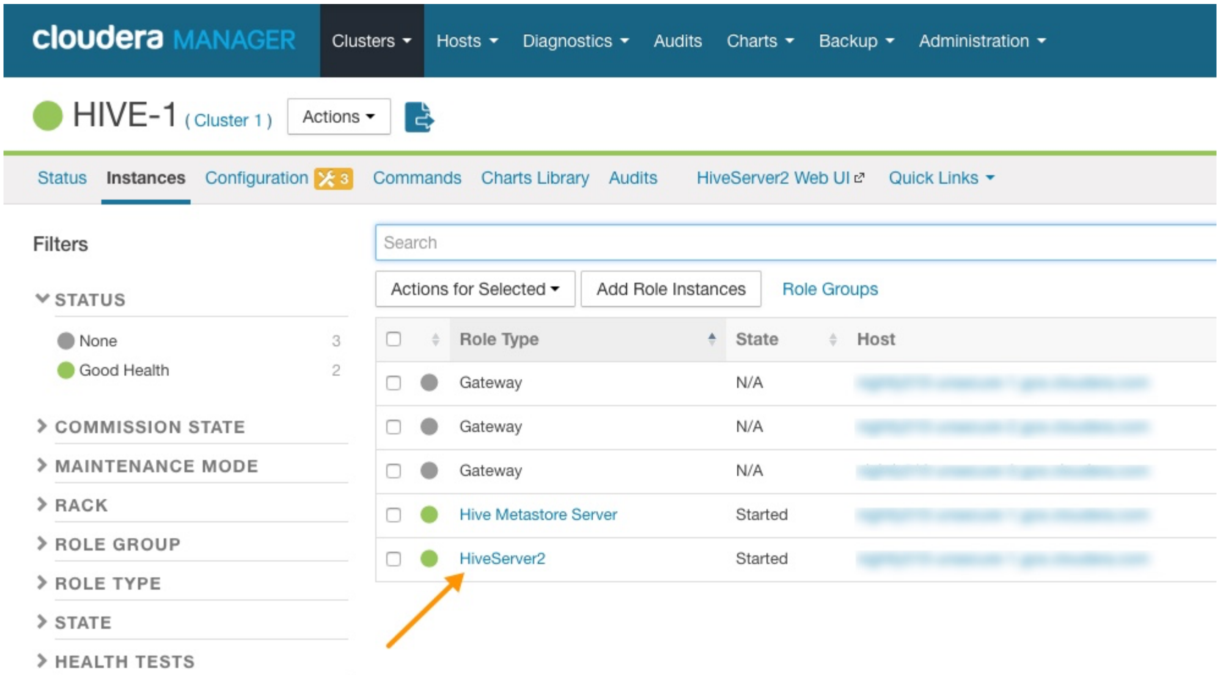


Figure 7: HiveServer2 Link on the Cloudera Manager Instances Page

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

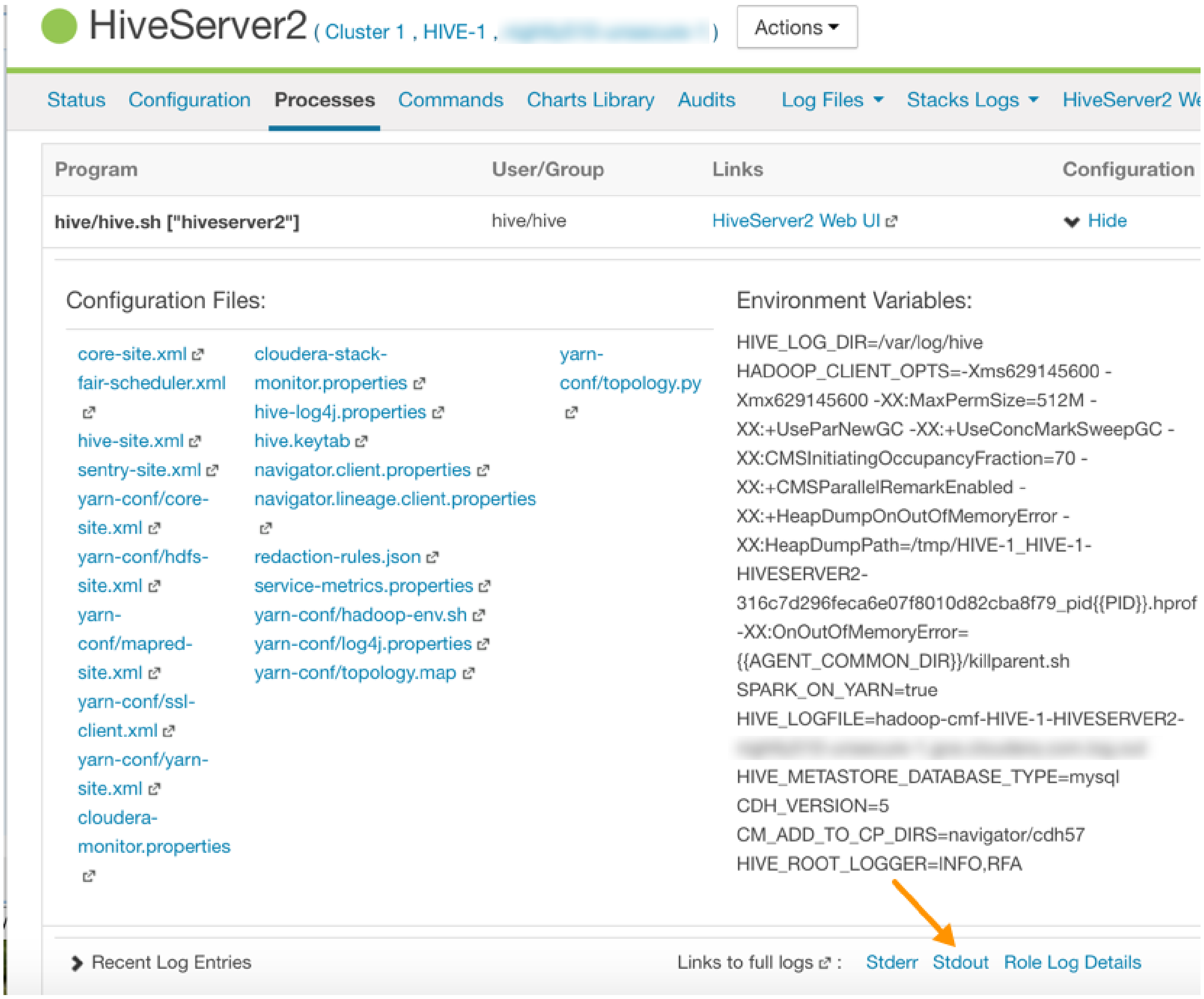


Figure 8: Link to the Stdout Log on the Cloudera Manager Processes Page

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

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

Video: Troubleshooting HiveServer2 Service Crashes

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

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

HiveServer2 General Performance Problems or Connections Refused

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

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

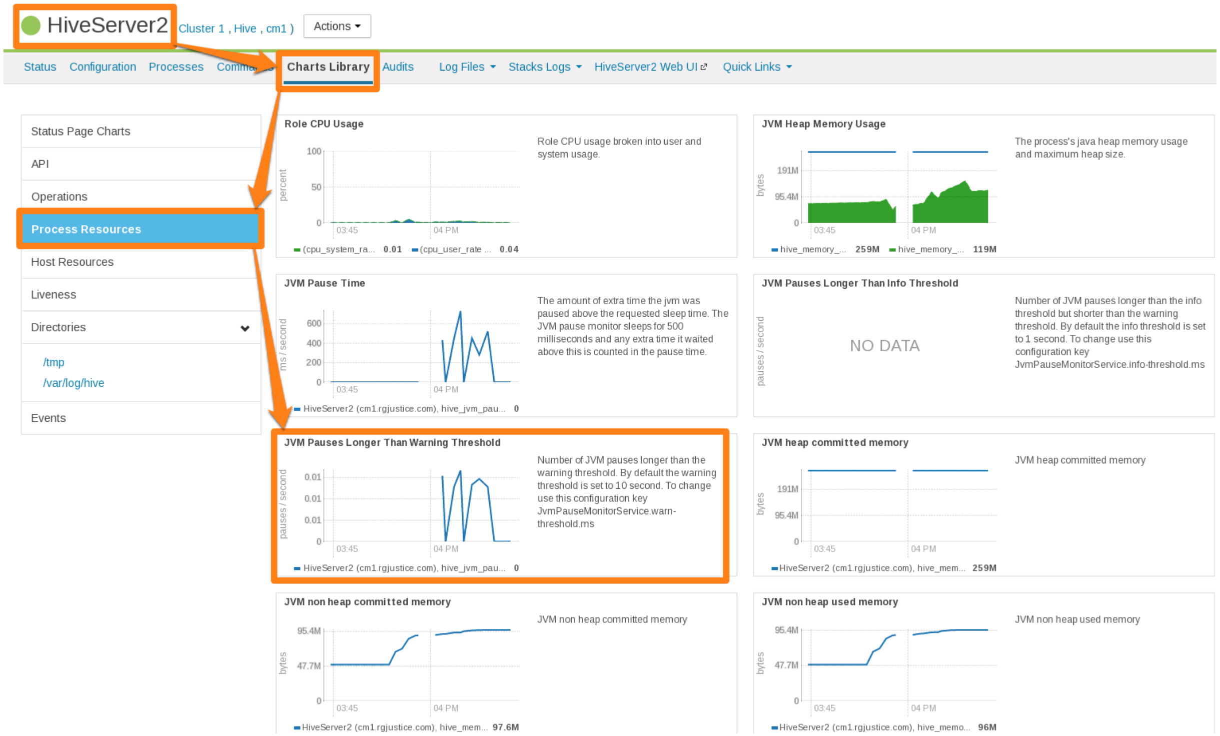


Figure 9: Cloudera Manager Chart Library Page for Process Resources

HiveServer2 Performance Best Practices

High heap usage by the HS2 process can be caused by Hive queries accessing high numbers of table partitions (greater than several thousand), high levels of concurrency, or other Hive workload characteristics described in Identify Workload Characteristics That Increase Memory Pressure on page 289.

HiveServer2 Heap Size Configuration Best Practices

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

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

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

   If HS2 is already configured to run with 4 GB or greater heap size and there are still performance issues, workload characteristics may be causing memory pressure. Increase heap size to reduce memory pressure on HS2. Cloudera does not recommend exceeding 16 GB per instance because of long garbage collection pause times. See Identify Workload Characteristics That Increase Memory Pressure on page 289 for tips to optimize query workloads to reduce the memory requirements on HS2. Cloudera recommends splitting HS2 into multiple instances and load-balancing once you start allocating over 16 GB to HS2.

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

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

- Reduce the number of services running on the HS2 host.
- Load-balance workloads across multiple HS2 instances as described in How the Number of Concurrent Connections Affect HiveServer2 Performance on page 289.
- Add more physical memory to the host or upgrade to a larger server.

How the Number of Concurrent Connections Affect HiveServer2 Performance

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

- **High number of concurrent queries**

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

  How to resolve:

  – Load-balance workloads across multiple HS2 instances by using HS2 load balancing, which is available in CDH 5.7 and later. Cloudera recommends that you determine the total number of HS2 servers on a cluster by dividing the expected maximum number of concurrent users on a cluster by 40. For example, if 400 concurrent users are expected, 10 HS2 instances should be available to support them. See Configuring HiveServer2 High Availability in CDH on page 429 for setup instructions.
  – Review usage patterns, such as batch jobs timing or Oozie workflows, to identify spikes in the number of connections that can be spread over time.

- **Many abandoned Hue sessions**

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

  How to resolve:

  – Reduce the session timeout duration for HS2, which minimizes the impact of abandoned Hue sessions. To reduce session timeout duration, modify these configuration parameters as follows:
    – `hive.server2.idle.operation.timeout=7200000`

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

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

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

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

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

Identify Workload Characteristics That Increase Memory Pressure

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

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

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

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

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

How to resolve:

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

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

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

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

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

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

  How to resolve:

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

- **High query complexity**

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

  How to resolve:

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

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

Improperly written UDFs can exert significant memory pressure on HS2.

How to resolve:

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

General Best Practices

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

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

Optimizing Hive Write Performance on Amazon S3

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

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

Configuring Optimized Hive S3 Write Performance

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

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

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

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

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

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

Setting Parameters on a Per-Query Basis

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

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

Setting Parameters as Service-Wide Defaults with Cloudera Manager

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

1. In the Cloudera Manager Admin Console, go to the Hive service.
2. Click the **Configuration** tab.
3. Click the **HiveServer2** scope.
4. Click the **Advanced** category.
5. Scroll down to the **HiveServer2 Advanced Configuration Snippet (Safety Valve)** for `hive-site.xml` and click the plus sign under the last configuration parameter in this section.

6. In the new **Name** and **Value** fields, type the **hive.blobstore.use.blobstore.as.scratchdir** parameter name and the **false** value, respectively.
7. Click the plus sign again, and in the **Name** and **Value** fields, type the **hive.exec.copyfile.maxsize** parameter name and the **1099511627776** value.
8. Click **Save Changes**.

Restrictions

The following restrictions apply to this feature:

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

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

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

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

- **HDFS Encryption**

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

Tuning Hive on Spark

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

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

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

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

YARN Configuration

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

Configuring Cores

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

Configuring Memory

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

For more information on tuning YARN, see Tuning YARN on page 310.

Spark Configuration

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

Configuring Executor Memory

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

When setting executor memory size, consider the following factors:

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

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

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

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

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

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

Configuring Driver Memory

You must also configure Spark driver memory:

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

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

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

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

Choosing the Number of Executors

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

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

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

Dynamic Executor Allocation

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

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

Parallelism

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

For more information on tuning Spark applications, see Tuning Spark Applications on page 304.

Hive Configuration

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

The size of data is described by two statistics:

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

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

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

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

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

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

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

Pre-warming YARN Containers

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

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

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

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

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

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

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

Tuning the Solr Server

Solr performance tuning is a complex task. The following sections provide more details.

Setting Java System Properties for Solr

Several of the following sections refer to Java system properties. These properties are set differently depending on whether or not you are using Cloudera Manager.

To set Java system properties for Solr in Cloudera Manager:

1. **Solr service** > **Configuration** > **Category** > **Advanced**
2. Add the property to **Java Configuration Options for Solr Server** using the format `-D<property_name>=<value>`. For example, to set `solr.hdfs.blockcache.slab.count` to `100`, add the following:

```
-Dsolr.hdfs.blockcache.slab.count=100
```

Garbage collection options, such as `-XX:+PrintGCTimeStamps`, can also be set here. Use spaces to separate multiple parameters.

To set Java system properties in unmanaged environments:

Add or modify the `CATALINA_OPTS` environment variable in `/etc/default/solr`. For example:

```
CATALINA_OPTS="-Xmx10g -XX:MaxDirectMemorySize=20g \
-XX:+UseLargePages -Dsolr.hdfs.blockcache.slab.count=100" \
-XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+PrintGCDetails
```

Tuning to Complete During Setup

Some tuning is best completed during the setup of you 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.

Configuring the Heap Size

Set the Java heap size for the Solr Server to at least 16 GB for production environments. For more information on memory requirements, see Guidelines for Deploying Cloudera Search.

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 batchSize 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=`.

For Cloudera Manager environments, you can set these flags at **Solr service** > **Configuration** > **Category** > **Java Configuration Options for Solr Server**.

For unmanaged environments, you can configure Java options by adding or modifying the `CATALINA_OPTS` environment variable in `/etc/default/solr`:

```
CATALINA_OPTS="-Xmx10g -XX:MaxDirectMemorySize=20g \
-XX:+UseLargePages -Dsolr.hdfs.blockcache.slab.count=100" \
-XX:+PrintGCTimeStamps -XX:+PrintGCDateStamps -XX:+PrintGCDetails
```

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 96 GB of memory, allocate 10-20 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 Java system property or by editing `solrconfig.xml` directly. For more information on setting Java system properties, see Setting Java System Properties for Solr on page 296.

If the parameters are set at the collection level (using `solrconfig.xml`), the first collection loaded by the Solr server takes precedence, and block cache settings in all other collections are ignored. Because you cannot control the order in which collections are loaded, you must make sure to set identical block cache settings in every collection `solrconfig.xml`. Block cache parameters set at the collection level in `solrconfig.xml` also take precedence over parameters at the process level.

Parameter	Cloudera Manager Setting	Default	Description
`solr.hdfs.blockcache.global`	Not directly configurable. Cloudera Manager automatically enables the global block cache. To override this setting, you must use the **Solr Service Environment**	true	If enabled, one HDFS block cache is used for each collection on a host. If `blockcache.global` is disabled, each SolrCore on a host creates its own private HDFS block cache. Enabling this parameter simplifies managing HDFS block cache memory.

Parameter	Cloudera Manager Setting	Default	Description
	Advanced Configuration Snippet (Safety Valve).		
solr.hdfs.blockcache.enabled	**HDFS Block Cache**	true	Enable the block cache.
solr.hdfs.blockcache.read.enabled	Not directly configurable. If the block cache is enabled, Cloudera Manager automatically enables the read cache. To override this setting, you must use the **Solr Service Environment Advanced Configuration Snippet (Safety Valve)**.	true	Enable the read cache.
solr.hdfs.blockcache.write.enabled	Not directly configurable. If the block cache is enabled, Cloudera Manager automatically disables the write cache. **Warning:** Do not enable the Solr HDFS write cache, because it can lead to index corruption.	false	Enable the write cache.
solr.hdfs.blockcache.direct.memory.allocation	**HDFS Block Cache Off-Heap Memory**	true	Enable direct memory allocation. If this is false, heap is used.
solr.hdfs.blockcache.blocksperbank	**HDFS Block Cache Blocks per Slab**	16384	Number of blocks per cache slab. The size of the cache is 8 KB (the block

Parameter	Cloudera Manager Setting	Default	Description
			size) times the number of blocks per slab times the number of slabs.
`solr.hdfs.blockcache.slab.count`	**HDFS Block Cache Number of Slabs**	1	Number of slabs per block cache. The size of the cache is 8 KB (the block size) times the number of blocks per slab times the number of slabs.

Note:

Increasing the direct memory cache size may make it necessary to increase the maximum direct memory size allowed by the JVM. Each Solr slab allocates 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 `MaxDirectMemorySize` to the number of slabs multiplied by the slab size does not provide enough memory.

To set `MaxDirectMemorySize` using Cloudera Manager:

1. Go to the Solr service.
2. Click the **Configuration** tab.
3. In the Search box, type **Java Direct Memory Size of Solr Server in Bytes**.
4. Set the new direct memory value.
5. Restart Solr servers after editing the parameter.

To set `MaxDirectMemorySize` in unmanaged environments:

1. Add `-XX:MaxDirectMemorySize=20g` to the `CATALINA_OPTS` environment variable in `/etc/default/solr`.
2. Restart Solr servers:

```
sudo service solr-server restart
```

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 <= `solr.hdfs.nrtcachingdirectory.maxmergesizemb`.
- The total cached bytes is <= `solr.hdfs.nrtcachingdirectory.maxcachedmb`.

The following parameters control NRT caching behavior:

Parameter	Default	Description
`solr.hdfs.nrtcachingdirectory.enable`	true	Whether to enable the NRTCachingDirectory.
`solr.hdfs.nrtcachingdirectory.maxcachedmb`	192	Size of the cache in megabytes.
`solr.hdfs.nrtcachingdirectory.maxmergesizemb`	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.slab.count">${solr.hdfs.blockcache.slab.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
```

- Disable swap space until next reboot:

```
sudo swapoff -a
```

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 the Java system property `-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 the Java system property `-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.

For information on setting Java system properties, see Setting Java System Properties for Solr on page 296.

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.

Replication

You can adjust the degree to which different data is replicated.

Replication Settings

> **Note:** Do not adjust HDFS replication settings for Solr in most cases.

To adjust the Solr replication factor for index files stored in HDFS:

- **Cloudera Manager:**
  1. Go to **Solr service** > **Configuration** > **Category** > **Advanced**.
  2. Click the plus sign next to **Solr Service Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** to add a new property with the following values:

     Name: `dfs.replication`

     Value: 2
  3. Click **Save Changes**.
  4. Restart the Solr service (**Solr service** > **Actions** > **Restart**).
- **Unmanaged:**
  1. Configure the `solr.hdfs.confdir` system property to refer to the Solr HDFS configuration files. Typically the value is `/etc/solrhdfs/`. For information on setting Java system properties, see Setting Java System Properties for Solr on page 296.
  2. Set the DFS replication value in the HDFS configuration file at the location you specified in the previous step. For example, to set the replication value to 2, you would change the `dfs.replication` setting as follows:

```
<property>
  <name>dfs.replication<name>
  <value>2<value>
<property>
```

  3. Restart the Solr service:

```
sudo service solr-server restart
```

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 supports configurable transaction log replication levels for replication logs stored in HDFS. Cloudera recommends leaving the value unchanged at 3 or, barring that, setting it to at least 2.

Configure the transaction log replication factor for a collection 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">${solr.ulog.dir:}</str>
     <int name="tlogDfsReplication">${solr.ulog.tlogDfsReplication:3}</int>
     <int name="numVersionBuckets">${solr.ulog.numVersionBuckets:65536}</int>
   </updateLog>
```

The default replication level is 3. For clusters with fewer than three DataNodes (such as proof-of-concept clusters), reduce this number to the amount of DataNodes in the cluster. Changing the replication level only applies to new transaction logs.

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

This topic describes various aspects in tuning Spark applications. During tuning you should monitor application behavior to determine the effect of tuning actions.

For information on monitoring Spark applications, see Monitoring Spark Applications .

Shuffle Overview

A Spark dataset comprises a fixed number of partitions, each of which comprises a number of records. For the datasets returned by ***narrow*** transformations, such as `map` and `filter`, the records required to compute the records in a single partition reside in a *single partition* in the parent dataset. Each object is only dependent on a single object in the parent. Operations such as `coalesce` can result in a task processing multiple input partitions, but the transformation is still considered narrow because the input records used to compute any single output record can still only reside in a limited subset of the partitions.

Spark also supports transformations with ***wide*** dependencies, such as `groupByKey` and `reduceByKey`. In these dependencies, the data required to compute the records in a single partition can reside in *many partitions* of the parent dataset. To perform these transformations, all of the tuples with the same key must end up in the same partition, processed by the same task. To satisfy this requirement, Spark performs a *shuffle*, which transfers data around the cluster and results in a new stage with a new set of partitions.

For example, consider the following code:

```
sc.textFile("someFile.txt").map(mapFunc).flatMap(flatMapFunc).filter(filterFunc).count()
```

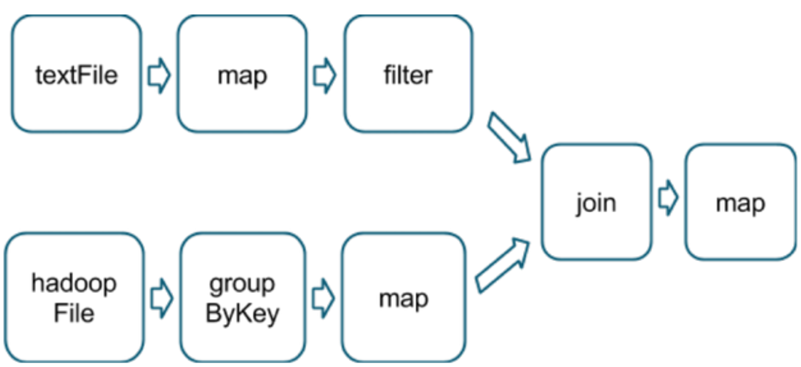
It runs a single action, `count`, which depends on a sequence of three transformations on a dataset derived from a text file. This code runs in a single stage, because none of the outputs of these three transformations depend on data that comes from different partitions than their inputs.

In contrast, this Scala code finds how many times each character appears in all the words that appear more than 1,000 times in a text file:

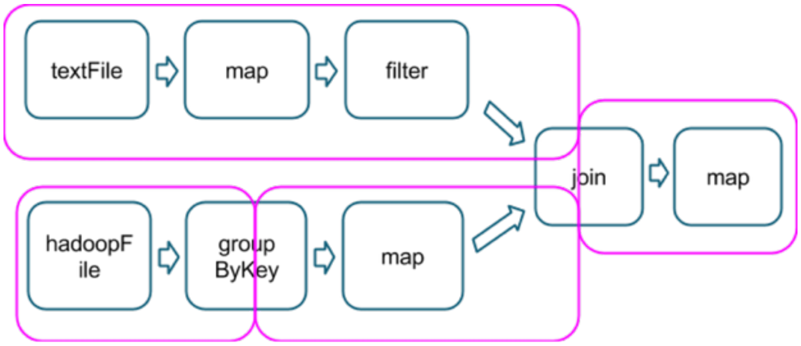
```
val tokenized = sc.textFile(args(0)).flatMap(_.split(' '))
val wordCounts = tokenized.map((_, 1)).reduceByKey(_ + _)
val filtered = wordCounts.filter(_._2 >= 1000)
val charCounts = filtered.flatMap(_._1.toCharArray).map((_, 1)).reduceByKey(_ + _)
charCounts.collect()
```

This example has three stages. The two `reduceByKey` transformations each trigger stage boundaries, because computing their outputs requires repartitioning the data by keys.

A final example is this more complicated transformation graph, which includes a `join` transformation with multiple dependencies:



The pink boxes show the resulting stage graph used to run it:



At each stage boundary, data is written to disk by tasks in the parent stages and then fetched over the network by tasks in the child stage. Because they incur high disk and network I/O, stage boundaries can be expensive and should be avoided when possible. The number of data partitions in a parent stage may be different than the number of partitions in a child stage. Transformations that can trigger a stage boundary typically accept a `numPartitions` argument, which specifies into how many partitions to split the data in the child stage. Just as the number of reducers is an important parameter in MapReduce jobs, the number of partitions at stage boundaries can determine an application's performance. Tuning the Number of Partitions on page 309 describes how to tune this number.

Choosing Transformations to Minimize Shuffles

You can usually choose from many arrangements of actions and transformations that produce the same results. However, not all these arrangements result in the same performance. Avoiding common pitfalls and picking the right arrangement can significantly improve an application's performance.

When choosing an arrangement of transformations, minimize the number of shuffles and the amount of data shuffled. Shuffles are expensive operations; all shuffle data must be written to disk and then transferred over the network. `repartition` , `join`, `cogroup`, and any of the `*By` or `*ByKey` transformations can result in shuffles. Not all these transformations are equal, however, and you should avoid the following patterns:

- `groupByKey` when performing an associative reductive operation. For example, `rdd.groupByKey().mapValues(_.sum)` produces the same result as `rdd.reduceByKey(_ + _)`. However, the former transfers the entire dataset across the network, while the latter computes local sums for each key in each partition and combines those local sums into larger sums after shuffling.
- `reduceByKey` when the input and output value types are *different*. For example, consider writing a transformation that finds all the unique strings corresponding to each key. You could use `map` to transform each element into a `Set` and then combine the `Set`s with `reduceByKey`:

```
rdd.map(kv => (kv._1, new Set[String]() + kv._2)).reduceByKey(_ ++ _)
```

  This results in unnecessary object creation because a new set must be allocated for each record.

  Instead, use `aggregateByKey`, which performs the map-side aggregation more efficiently:

```
val zero = new collection.mutable.Set[String]()
rdd.aggregateByKey(zero)((set, v) => set += v,(set1, set2) => set1 ++= set2)
```

- `flatMap-join-groupBy`. When two datasets are already grouped by key and you want to join them and keep them grouped, use `cogroup`. This avoids the overhead associated with unpacking and repacking the groups.

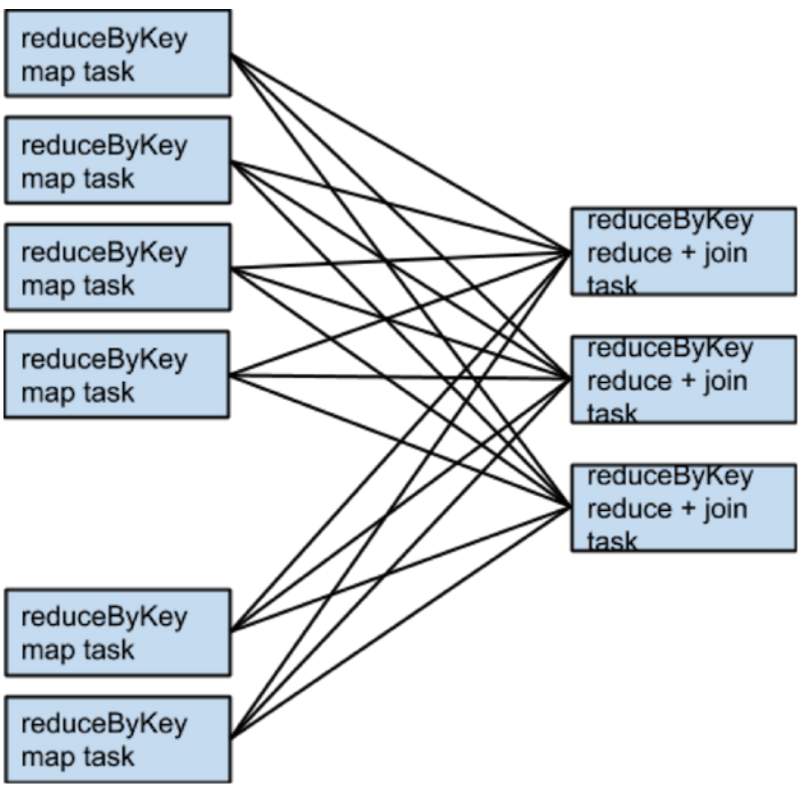
When Shuffles Do Not Occur

In some circumstances, the transformations described previously *do not* result in shuffles. Spark does not shuffle when a previous transformation has already partitioned the data according to the *same partitioner*. Consider the following flow:

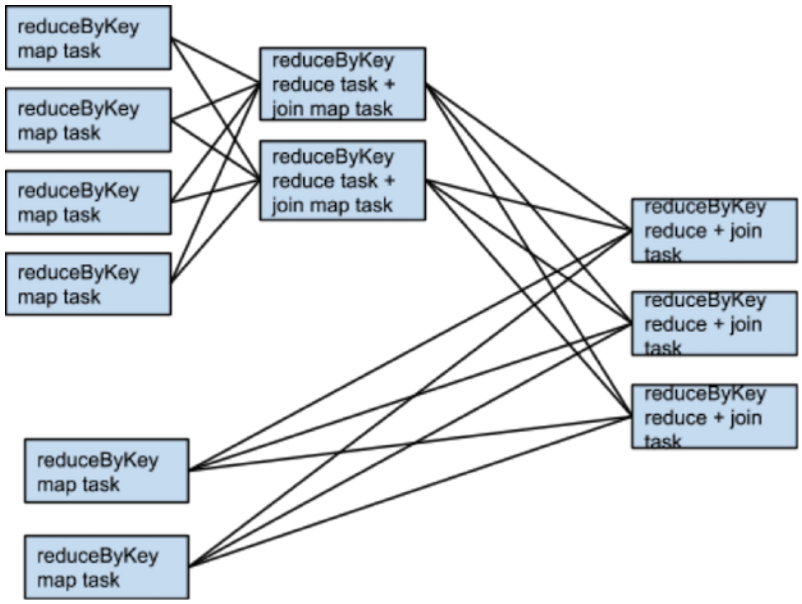
```
rdd1 = someRdd.reduceByKey(...)
rdd2 = someOtherRdd.reduceByKey(...)
rdd3 = rdd1.join(rdd2)
```

Because no partitioner is passed to `reduceByKey`, the default partitioner is used, resulting in `rdd1` and `rdd2` both being hash-partitioned. These two `reduceByKey` transformations result in two shuffles. If the datasets have the same number of partitions, a join requires no additional shuffling. Because the datasets are partitioned identically, the set of keys in any single partition of `rdd1` can only occur in a single partition of `rdd2`. Therefore, the contents of any single output partition of `rdd3` depends only on the contents of a single partition in `rdd1` and single partition in `rdd2`, and a third shuffle is not required.

For example, if `someRdd` has four partitions, `someOtherRdd` has two partitions, and both the `reduceByKeys` use three partitions, the set of tasks that run would look like this:



If `rdd1` and `rdd2` use different partitioners or use the default (hash) partitioner with different numbers of partitions, only one of the datasets (the one with the fewer number of partitions) needs to be reshuffled for the join:



To avoid shuffles when joining two datasets, you can use broadcast variables. When one of the datasets is small enough to fit in memory in a single executor, it can be loaded into a hash table on the driver and then broadcast to every executor. A map transformation can then reference the hash table to do lookups.

When to Add a Shuffle Transformation

The rule of minimizing the number of shuffles has some exceptions.

An extra shuffle can be advantageous when it increases parallelism. For example, if your data arrives in a few large unsplittable files, the partitioning dictated by the `InputFormat` might place large numbers of records in each partition, while not generating enough partitions to use all available cores. In this case, invoking repartition with a high number of partitions (which triggers a shuffle) after loading the data allows the transformations that follow to use more of the cluster's CPU.

Another example arises when using the `reduce` or `aggregate` action to aggregate data into the driver. When aggregating over a high number of partitions, the computation can quickly become bottlenecked on a single thread in the driver merging all the results together. To lighten the load on the driver, first use `reduceByKey` or `aggregateByKey` to perform a round of distributed aggregation that divides the dataset into a smaller number of partitions. The values in each partition are merged with each other in parallel, before being sent to the driver for a final round of aggregation. See `treeReduce` and `treeAggregate` for examples of how to do that.

This method is especially useful when the aggregation is already grouped by a key. For example, consider an application that counts the occurrences of each word in a corpus and pulls the results into the driver as a map. One approach, which can be accomplished with the `aggregate` action, is to compute a local map at each partition and then merge the maps at the driver. The alternative approach, which can be accomplished with `aggregateByKey`, is to perform the count in a fully distributed way, and then simply `collectAsMap` the results to the driver.

Secondary Sort

The `repartitionAndSortWithinPartitions` transformation repartitions the dataset according to a partitioner and, within each resulting partition, sorts records by their keys. This transformation pushes sorting down into the shuffle machinery, where large amounts of data can be spilled efficiently and sorting can be combined with other operations.

For example, Apache Hive on Spark uses this transformation inside its `join` implementation. It also acts as a vital building block in the *secondary sort* pattern, in which you group records by key and then, when iterating over the values that correspond to a key, have them appear in a particular order. This scenario occurs in algorithms that need to group events by user and then analyze the events for each user, based on the time they occurred.

Tuning Resource Allocation

For background information on how Spark applications use the YARN cluster manager, see [Running Spark Applications on YARN](#).

The two main resources that Spark and YARN manage are CPU and memory. Disk and network I/O affect Spark performance as well, but neither Spark nor YARN actively manage them.

Every Spark executor in an application has the same fixed number of cores and same fixed heap size. Specify the number of cores with the `--executor-cores` command-line flag, or by setting the `spark.executor.cores` property. Similarly, control the heap size with the `--executor-memory` flag or the `spark.executor.memory` property. The `cores` property controls the number of concurrent tasks an executor can run. For example, set `--executor-cores 5` for each executor to run a maximum of five tasks at the same time. The memory property controls the amount of data Spark can cache, as well as the maximum sizes of the shuffle data structures used for grouping, aggregations, and joins.

Starting with CDH 5.5 [dynamic allocation](#), which adds and removes executors dynamically, is enabled. To explicitly control the number of executors, you can override dynamic allocation by setting the `--num-executors` command-line flag or `spark.executor.instances` configuration property.

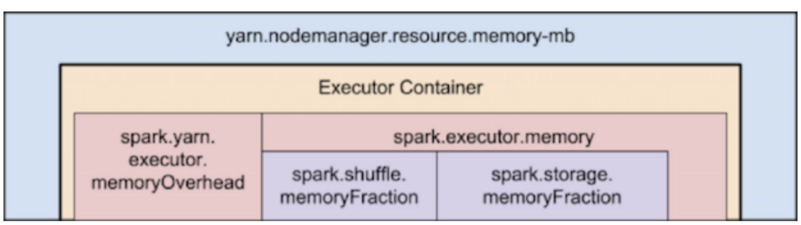
Consider also how the resources requested by Spark fit into resources YARN has available. The relevant YARN properties are:

- `yarn.nodemanager.resource.memory-mb` controls the maximum sum of memory used by the containers on each host.
- `yarn.nodemanager.resource.cpu-vcores` controls the maximum sum of cores used by the containers on each host.

Requesting five executor cores results in a request to YARN for five cores. The memory requested from YARN is more complex for two reasons:

- The `--executor-memory/spark.executor.memory` property controls the executor heap size, but executors can also use some memory off heap, for example, Java NIO direct buffers. The value of the `spark.yarn.executor.memoryOverhead` property is added to the executor memory to determine the full memory request to YARN for each executor. It defaults to max(384, .1 * `spark.executor.memory`).
- YARN may round the requested memory up slightly. The `yarn.scheduler.minimum-allocation-mb` and `yarn.scheduler.increment-allocation-mb` properties control the minimum and increment request values, respectively.

The following diagram (not to scale with defaults) shows the hierarchy of memory properties in Spark and YARN:



Keep the following in mind when sizing Spark executors:

- The ApplicationMaster, which is a non-executor container that can request containers from YARN, requires memory and CPU that must be accounted for. In ***client*** deployment mode, they default to 1024 MB and one core. In ***cluster*** deployment mode, the ApplicationMaster runs the driver, so consider bolstering its resources with the `--driver-memory` and `--driver-cores` flags.
- Running executors with too much memory often results in excessive garbage-collection delays. For a single executor, use 64 GB as an upper limit.
- The HDFS client has difficulty processing many concurrent threads. At most, five tasks per executor can achieve full write throughput, so keep the number of cores per executor below that number.

- Running tiny executors (with a single core and just enough memory needed to run a single task, for example) offsets the benefits of running multiple tasks in a single JVM. For example, broadcast variables must be replicated once on each executor, so many small executors results in many more copies of the data.

Resource Tuning Example

Consider a cluster with six hosts running NodeManagers, each equipped with 16 cores and 64 GB of memory.

The NodeManager capacities, `yarn.nodemanager.resource.memory-mb` and `yarn.nodemanager.resource.cpu-vcores`, should be set to 63 * 1024 = 64512 (megabytes) and 15, respectively. Avoid allocating 100% of the resources to YARN containers because the host needs some resources to run the OS and Hadoop daemons. In this case, leave one GB and one core for these system processes. Cloudera Manager accounts for these and configures these YARN properties automatically.

You might consider using `--num-executors 6 --executor-cores 15 --executor-memory 63G`. However, this approach does not work:

- 63 GB plus the executor memory overhead does not fit within the 63 GB capacity of the NodeManagers.
- The ApplicationMaster uses a core on one of the hosts, so there is no room for a 15-core executor on that host.
- 15 cores per executor can lead to bad HDFS I/O throughput.

Instead, use `--num-executors 17 --executor-cores 5 --executor-memory 19G`:

- This results in three executors on all hosts except for the one with the ApplicationMaster, which has two executors.
- `--executor-memory` is computed as (63/3 executors per host) = 21. 21 * 0.07 = 1.47. 21 - 1.47 ~ 19.

Tuning the Number of Partitions

Spark has limited capacity to determine optimal parallelism. Every Spark stage has a number of tasks, each of which processes data sequentially. The number of tasks per stage is the most important parameter in determining performance.

As described in Spark Execution Model, Spark groups datasets into stages. The number of tasks in a stage is the same as the number of partitions in the last dataset in the stage. The number of partitions in a dataset is the same as the number of partitions in the datasets on which it depends, with the following exceptions:

- The `coalesce` transformation creates a dataset with *fewer* partitions than its parent dataset.
- The `union` transformation creates a dataset with the *sum* of its parents' number of partitions.
- The `cartesian` transformation creates a dataset with the *product* of its parents' number of partitions.

Datasets with no parents, such as those produced by `textFile` or `hadoopFile`, have their partitions determined by the underlying MapReduce `InputFormat` used. Typically, there is a partition for each HDFS block being read. The number of partitions for datasets produced by `parallelize` are specified in the method, or `spark.default.parallelism` if not specified. To determine the number of partitions in an dataset, call `rdd.partitions().size()`.

If the number of tasks is smaller than number of slots available to run them, CPU usage is suboptimal. In addition, more memory is used by any aggregation operations that occur in each task. In `join`, `cogroup`, or `*ByKey` operations, objects are held in hashmaps or in-memory buffers to group or sort. `join`, `cogroup`, and `groupByKey` use these data structures in the tasks for the stages that are on the fetching side of the shuffles they trigger. `reduceByKey` and `aggregateByKey` use data structures in the tasks for the stages on both sides of the shuffles they trigger. If the records in these aggregation operations exceed memory, the following issues can occur:

- Increased garbage collection, which can lead to pauses in computation.
- Spilling data to disk, causing disk I/O and sorting, which leads to job stalls.

To increase the number of partitions if the stage is reading from Hadoop:

- Use the `repartition` transformation, which triggers a shuffle.
- Configure your `InputFormat` to create more splits.
- Write the input data to HDFS with a smaller block size.

If the stage is receiving input from another stage, the transformation that triggered the stage boundary accepts a `numPartitions` argument:

```
val rdd2 = rdd1.reduceByKey(_ + _, numPartitions = X)
```

Determining the optimal value for *X* requires experimentation. Find the number of partitions in the parent dataset, and then multiply that by 1.5 until performance stops improving.

You can also calculate *X* using a formula, but some quantities in the formula are difficult to calculate. The main goal is to run enough tasks so that the data destined for each task fits in the memory available to that task. The memory available to each task is:

```
(spark.executor.memory * spark.shuffle.memoryFraction * spark.shuffle.safetyFraction)/
spark.executor.cores
```

`memoryFraction` and `safetyFraction` default to 0.2 and 0.8 respectively.

The in-memory size of the total shuffle data is more difficult to determine. The closest heuristic is to find the ratio between shuffle spill memory and the shuffle spill disk for a stage that ran. Then, multiply the total shuffle write by this number. However, this can be compounded if the stage is performing a reduction:

```
(observed shuffle write) * (observed shuffle spill memory) * (spark.executor.cores)/
(observed shuffle spill disk) * (spark.executor.memory) * (spark.shuffle.memoryFraction)
 * (spark.shuffle.safetyFraction)
```

Then, round up slightly, because too many partitions is usually better than too few.

When in doubt, err on the side of a larger number of tasks (and thus partitions). This contrasts with recommendations for MapReduce, which unlike Spark, has a high startup overhead for tasks.

Reducing the Size of Data Structures

Data flows through Spark in the form of records. A record has two representations: a deserialized Java object representation and a serialized binary representation. In general, Spark uses the deserialized representation for records in memory and the serialized representation for records stored on disk or transferred over the network. For sort-based shuffles, in-memory shuffle data is stored in serialized form.

The `spark.serializer` property controls the serializer used to convert between these two representations. Cloudera recommends using the Kryo serializer, `org.apache.spark.serializer.KryoSerializer`.

The footprint of your records in these two representations has a significant impact on Spark performance. Review the data types that are passed and look for places to reduce their size. Large deserialized objects result in Spark spilling data to disk more often and reduces the number of deserialized records Spark can cache (for example, at the `MEMORY` storage level). The Apache Spark tuning guide describes how to reduce the size of such objects. Large serialized objects result in greater disk and network I/O, as well as reduce the number of serialized records Spark can cache (for example, at the `MEMORY_SER` storage level.) Make sure to register any custom classes you use with the `SparkConf#registerKryoClasses` API.

Choosing Data Formats

When storing data on disk, use an extensible binary format like Avro, Parquet, Thrift, or Protobuf and store in a sequence file.

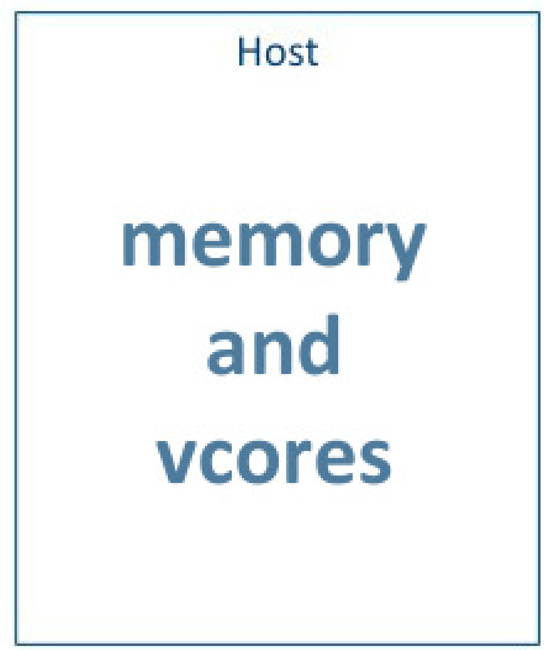
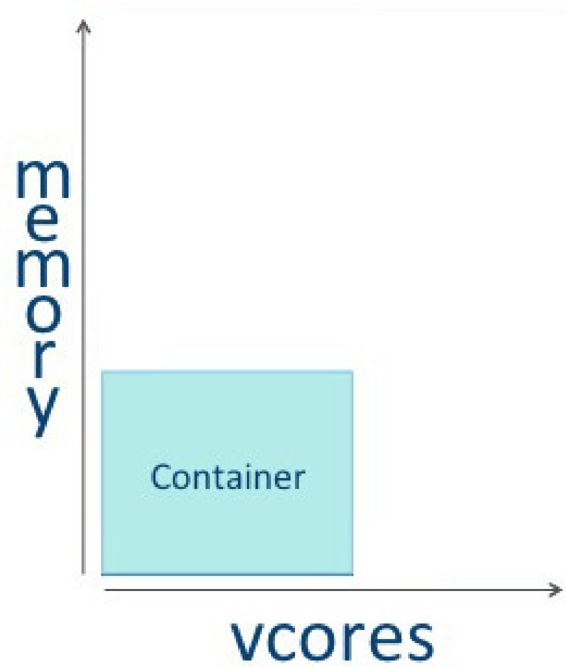
Tuning YARN

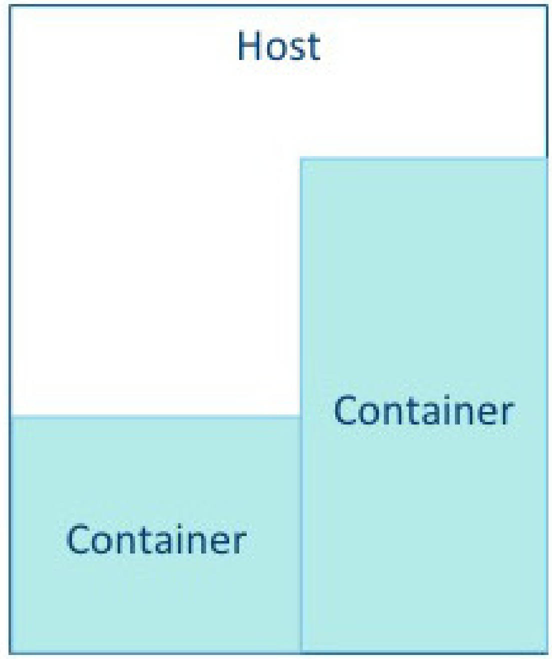
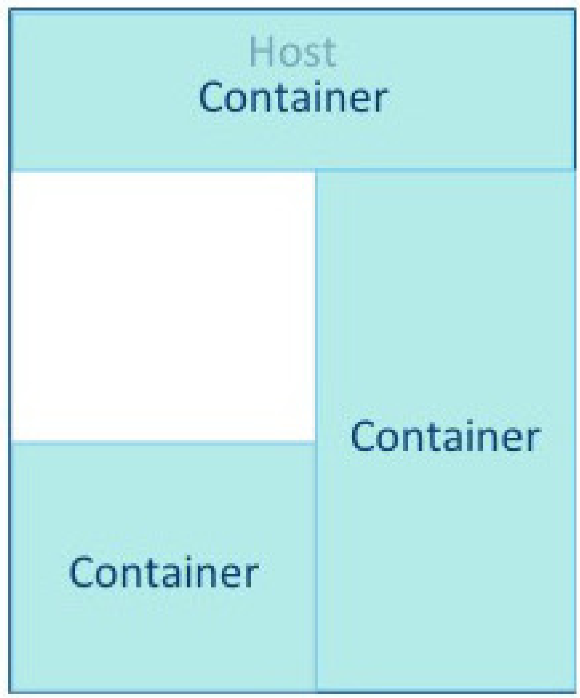
This topic applies to YARN clusters only, and describes how to tune and optimize YARN for your cluster.

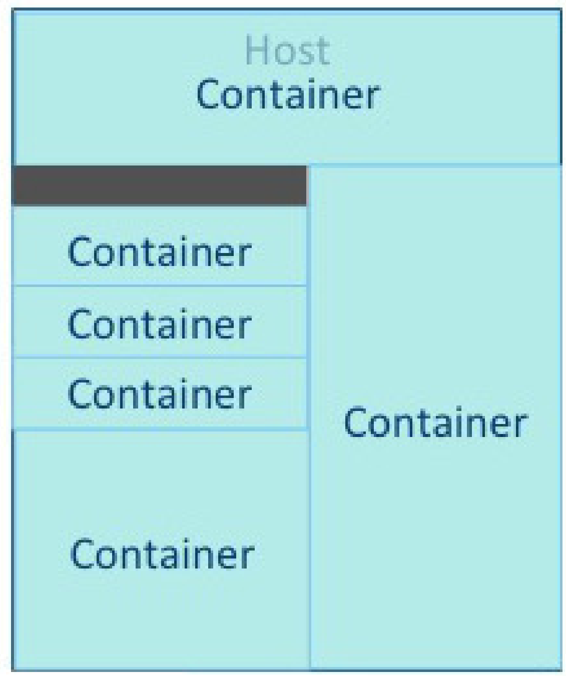
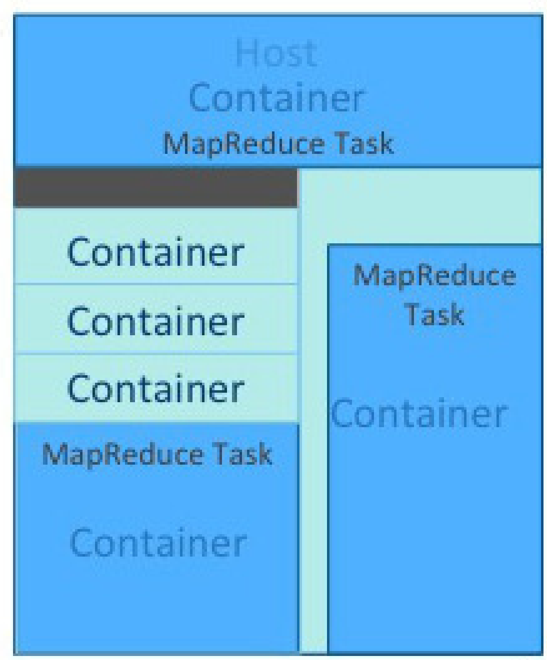
> **Note:** Download the Cloudera YARN tuning spreadsheet to help calculate YARN configurations. For a short video overview, see Tuning YARN Applications.

Overview

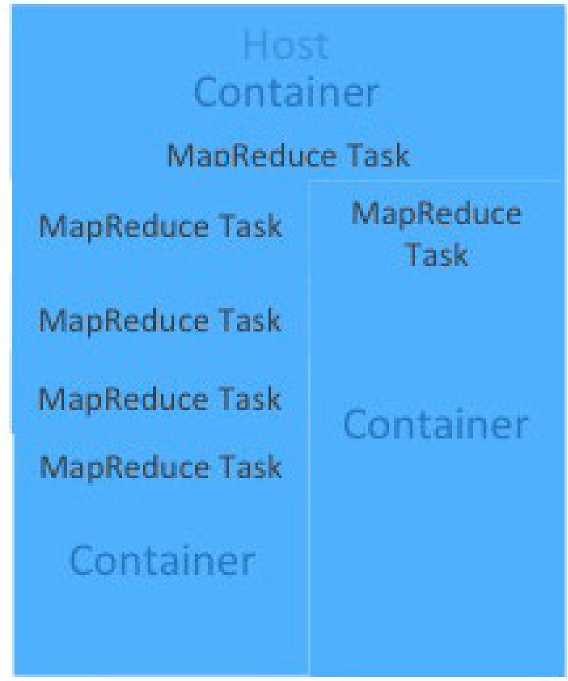
This overview provides an abstract description of a YARN cluster and the goals of YARN tuning.

A YARN cluster is composed of host machines. Hosts provide memory and CPU resources. A *vcore*, or virtual core, is a usage share of a host CPU.	
Tuning YARN consists primarily of optimally defining *containers* on your worker hosts. You can think of a container 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 ResourceManager allocates memory and vcores to use all available resources in the most efficient way possible. Ideally, few or no resources are left idle.	
An *application* is a YARN client program consisting of one or more tasks. Typically, a task uses all of the available resources in the container. A task cannot consume more than its designated allocation, ensuring that it cannot use all of the host CPU cycles or exceed its memory allotment.	

Tune your YARN hosts to optimize the use of vcores and memory by configuring your containers to use all available resources beyond those required for overhead and other services.



YARN tuning has three phases. 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.

YARN and MapReduce have many configurable properties. For a complete list, see 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.

Host Components	Quantity	Size	Total	Description / Notes
RAM	256G		256G	Node memory in Gigabytes
CPU	4	6	48	Number of CPU's and the number of HW cores per CPU. The calculation of vcores below includes HyperThreading support.
HyperThreading CPU	yes			Does the CPU support HyperThreading?
HDD (Hard Disk Drive)	24	3T	72G	Number of Hard Drives and size per drive in JBOD Configuration
Ethernet	2	1G	2G	Number of Ethernet connections and the transfer speed

As with any system, the more memory and CPU resources available, the faster the cluster can process large amounts of data. A machine with 4 CPUs with HyperThreading, each with 6 cores, provides 48 vcores per host.

3 TB 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. The cost of storage decreases over time, so you might consider 4 TB 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)	Notes
Operating System	Overhead	1	8192	Most operating systems use 4-8GB minimum.
Other services	Overhead	0	0	Enter the required cores or memory for non CDH services not part of the OS.
Cloudera Manager agent	Overhead	1	1024	Allocate 1GB and 1 vcore for Cloudera Manager agents, which track resource usage on a host.
HDFS DataNode	CDH	1	1024	Allocation for the HDFS DataNode heap: default 1GB and 1 vcore.
YARN NodeManager	CDH	1	1024	Allocation for the YARN NodeManager heap: default 1GB and 1 vcore
Impala daemon	CDH	0	0	(Optional Service) Suggestion: Allocate at least 16GB memory when using Impala.
Hbase RegionServer	CDH	0	0	(Optional Service) Suggestion: Allocate no more than 12-16GB memory when using HBase Region Servers.
Solr Server	CDH	0	0	(Optional Service) Suggestion: Minimum 1GB for Solr server. More might be necessary depending on index sizes.
Kudu Server	CDH	0	0	(Optional Service) Suggestion: Minimum 1GB for Kudu Tablet server. More might be necessary depending on data sizes.
Available Container Resources		44	250880	
Container resources				
Physical Cores to Vcores Multiplier		1		Set this ratio based on the expected number of concurrent threads in a container per thread core. Default is 1.
YARN Available Vcores		44		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 1 GB for Cloudera Manager. If services outside of CDH require additional resources, add those numbers under Other Services.

The HDFS DataNode uses a minimum of 1 core and about 1 GB of memory. The same requirements apply to the YARN NodeManager.

The spreadsheet lists several optional services:

- Impala daemon requires at least 16 GB for the daemon.
- HBase Region Servers requires 12-16 GB of memory.
- Solr server requires a minimum of 1 GB of memory.
- Kudu Tablet server requires a minimum of 1 GB of memory.

Any remaining resources are available for YARN applications (Spark and MapReduce). In this example, 44 CPU cores are 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. To see the benefits of parallel computing, set the number of hosts to a minimum of 10.

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

YARN NodeManager Configuration Properties	Value	Note
yarn.nodemanager.resource.cpu-vcores	44	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"	440	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 can change the 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 to the size of the node.

Set the minimum and maximum reservations for memory. The increment should be the smallest amount that can impact performance. Here, the minimum is approximately 1 GB, the 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 Properties (Vcores)	Value	Description
yarn.scheduler.minimum-allocation-vcores	1	Minimum vcore reservation for a container
yarn.scheduler.maximum-allocation-vcores	44	Maximum vcore reservation for a container
yarn.scheduler.increment-allocation-vcores	1	Vcore allocations must be a multiple of this value

YARN Container Configuration Properties (Memory)	Value	Description
yarn.scheduler.minimum-allocation-mb	1024	Minimum memory reservation for a container in MegaByte
yarn.scheduler.maximum-allocation-mb	250880	Maximum memory reservation for a container in MegaByte
yarn.scheduler.increment-allocation-mb	512	Memory allocations must be a multiple of this value in MegaByte

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	Minimum	Maximum
Max possible number of containers, based on memory configuration		2450
Max possible number of containers, based on vcore configuration		440
Container number based on 2 containers per disk spindles		480
Min possible number of containers, based on memory configuration	10	
Min possible number of containers, based on vcore configuration	10	

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
Scheduler maximum vcores must be larger than minimum	GOOD	yarn.scheduler.maximum-allocation-vcores >= yarn.scheduler.minimum-allocation-vcores
Scheduler maximum allocation MB must be larger than minimum	GOOD	yarn.scheduler.maximum-allocation-mb >= yarn.scheduler.minimum-allocation-mb
Scheduler minimum vcores must be greater than or equal to 0	GOOD	yarn.scheduler.minimum-allocation-vcores >= 0
Scheduler maximum vcores must be greater than or equal to 1	GOOD	yarn.scheduler.maximum-allocation-vcores >= 1
Host vcores must be larger than scheduler minimum vcores	GOOD	yarn.nodemanager.resource.cpu-vcores >= yarn.scheduler.minimum-allocation-vcores
Host vcores must be larger than scheduler maximum vcores	GOOD	yarn.nodemanager.resource.cpu-vcores >= yarn.scheduler.maximum-allocation-vcores
Host allocation MB must be larger than scheduler minimum	GOOD	yarn.nodemanager.resource.memory-mb >= yarn.scheduler.maximum-allocation-mb
Host allocation MB must be larger than scheduler maximum vcores	GOOD	yarn.nodemanager.resource.memory-mb >= yarn.scheduler.minimum-allocation-mb
Small container limit	GOOD	If yarn.scheduler.minimum-allocation-mb is less than 1GB, containers will likely get killed by YARN due to OutOfMemory issues

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 ApplicationMaster, map tasks, and reduce tasks. The minimum vcore allocation for any task is always 1. The Spill/Sort memory allocation of 400 should be sufficient, and should be (rarely) increased if you determine that frequent spills to disk are hurting job performance.

In CDH 5.5 and higher, the common MapReduce parameters `mapreduce.map.java.opts`, `mapreduce.reduce.java.opts`, and `yarn.app.mapreduce.am.command-opts` are configured for you automatically based on the *Heap to Container Size Ratio*.

STEP 7: MapReduce Configuration

For CDH 5.5 and later we recommend that only the heap or the container size is specified for map and reduce tasks. The value that is not specified will be calculated based on the setting mapreduce.job.heap.memory-mb.ratio. This calculation follows Cloudera Manager and calculates the heap size based on the ratio and the container size.

Application Master Configuration properties		Value	Description
yarn.app.mapreduce.am.resource.cpu-vcores		1	AM container vcore reservation
yarn.app.mapreduce.am.resource.mb		1024	AM container memory reservation in MegaByte
yarn.app.mapreduce.am.command-opts	-Xmx	800	AM Java heap size in MegaByte
Task auto heap sizing			
Use task auto heap sizing		yes	
mapreduce.job.heap.memory-mb.ratio		0.8	Ratio between the container size and task heap size.
Map Task Configuration properties			
mapreduce.map.cpu.vcores		1	Map task vcore reservation
mapreduce.map.memory.mb		1024	Map task memory reservation in MegaByte
mapreduce.map.java.opts	ignored	800	Map task Java heap size in MegaByte
mapreduce.task.io.sort.mb		400	Spill/Sort memory reservation
ReduceTask Configuration properties			
mapreduce.reduce.cpu.vcores		1	Reduce task vcore reservation
mapreduce.reduce.memory.mb		1024	Reduce task memory reservation in MegaByte
mapreduce.reduce.java.opts	ignored	800	Reduce Task Java heap size in MegaByte

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
AM vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **yarn.app.mapreduce.am.resource.cpu-vcores** <= yarn-scheduler.maximum-allocation-vcores
AM memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **yarn.app.mapreduce.am.resource.cpu-vcores** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	GOOD	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM
Ratio should be between 0.75 and 0.9	GOOD	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM

Map Task Sanity Checks	Value	Description
Map task vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **mapreduce.map.cpu.vcores** <= yarn-scheduler.maximum-allocation-vcores
Map task memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **mapreduce.map.memory.mb** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	N/A	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM
Spill/Sort memory should not use whole map task heap	GOOD	Make sure that Spill/Sort memory reservation uses between 40% and 60% of the heap of a map task.

Reduce Task Sanity Checks	Value	Description
Reduce task vcore request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-vcores <= **mapreduce.reduce.cpu.vcores** <= yarn-scheduler.maximum-allocation-vcores
Reduce task memory request must fit within scheduler limits	GOOD	yarn.scheduler.minimum-allocation-mb <= **mapreduce.reduce.memory.mb** <= yarn.scheduler.maximum-allocation-mb
Container size must large enough for java heap and overhead	N/A	Java Heap should be between 75% and 90% of the container size: too low wastes resources, to high could lead to OOM

Continuous Scheduling

Enabling or disabling continuous scheduling changes how often YARN schedules, either continuously or based on the node heartbeats. For larger clusters (more than 75 nodes) seeing heavy YARN workloads, disabling continuous scheduling with the following settings is recommended in general:

- `yarn.scheduler.fair.continuous-scheduling-enabled` should be false
- `yarn.scheduler.fair.assignmultiple` should be true

On large clusters, continuous scheduling can cause the ResourceManager to appear unresponsive since continuous scheduling iterates through all the nodes in the cluster.

For more information about continuous scheduling tuning, see the following knowledge base article: [FairScheduler Tuning with assignmultiple and Continuous Scheduling]

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 Using Cloudera Manager on page 13.

Table 17: 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

Resource Management

Resource management helps ensure predictable behavior by defining the impact of different services on cluster resources. Use resource management 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.

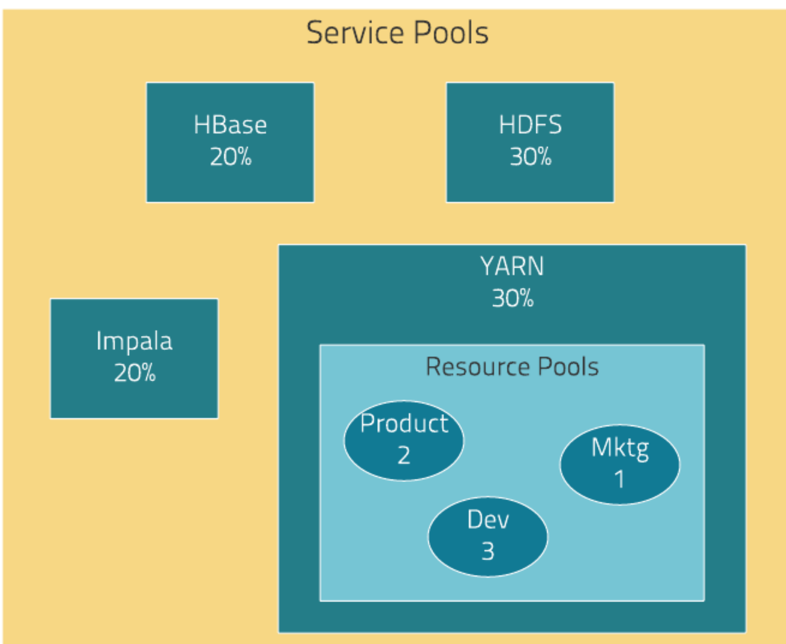
Cloudera Manager Resource Management

Cloudera Manager provides two methods for allocating cluster resources to services: static and dynamic.

Static Allocation

With Cloudera Manager 5, statically allocating resources using cgroups is configurable through a single *static service pool* wizard. You allocate services as 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

You can dynamically apportion resources that are statically allocated to YARN and Impala by using *dynamic resource pools*.

Depending on the version of CDH you are using, dynamic resource pools in Cloudera Manager support the following scenarios:

- **YARN (CDH 5)** - 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 receives 30% x 2/6 (or 10%) of the total cluster resources. If no applications are using the Dev and Mktg pools, the YARN Product pool is allocated 30% of the cluster resources.

- **Impala (CDH 5 and CDH 4)** - Impala manages memory for pools running queries and limits the number of running and queued queries in each pool.

The scenarios in which YARN manages resources map to the YARN scheduler policy. The scenarios in which Impala independently manages resources use Impala admission control.

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*** > **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*** > **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%. 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.
6. After you enable static service pools, there are three additional tasks.
   a. Delete everything under the local directory path on NodeManager hosts. The local directory path is configurable, and can be verified in Cloudera Manager with **YARN** > **Configuration** > **NodeManager Local Directories**.
   b. Enable cgroups for resource management. You can enable cgroups in Cloudera Manager with **Yarn** > **Configuration** > **Use CGroups for Resource Management**.
   c. If you are using the optional Impala scratch directory, delete all files in the Impala scratch directory. The directory path is configurable, and can be verified in Cloudera Manager with **Impala** > **Configuration** > **Impala Daemon Scratch Directories**.

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. Clear 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 35.

Linux Control Groups (cgroups)

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 tables. If a distribution lacks support for a given parameter, changes to the parameter have no effect.

Table 18: RHEL-compatible

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
Red Hat Enterprise Linux, CentOS, and Oracle Enterprise Linux 7	■	■	■	■

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
Red Hat Enterprise Linux, CentOS, and Oracle Enterprise Linux 6	■	■	■	■
Red Hat Enterprise Linux, CentOS, and Oracle Enterprise Linux 5				

Table 19: SLES

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
SUSE Linux Enterprise Server 11	■	■	■	■

Table 20: Ubuntu

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
Ubuntu 14.04 LTS	■	■	■	■
Ubuntu 12.04 LTS	■	■	■	■
Ubuntu 10.04 LTS	■		■	■

Table 21: Debian

Distribution	CPU Shares	I/O Weight	Memory Soft Limit	Memory Hard Limit
Debian 7.1	■			
Debian 7.0	■			
Debian 6.0	■			

The exact level of support can be found in the Cloudera Manager Agent log file, shortly after the Agent has started. See Viewing the Cloudera Manager Server Log 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_cpu and similar entries correspond directly to support for the CPU, I/O, and memory parameters.

Further Reading

- https://www.kernel.org/doc/Documentation/cgroup-v1/cgroups.txt
- https://www.kernel.org/doc/Documentation/cgroup-v1/blkio-controller.txt
- https://www.kernel.org/doc/Documentation/cgroup-v1/memory.txt
- Managing System Resources on Red Hat Enterprise Linux 6
- Managing System Resources on Red Hat Enterprise Linux 7

Resource Management with Control Groups

To use cgroups, you must enable cgroup-based resource management under the host resource management configuration properties. However, if you configure static service pools, this property is 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:
      - 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
```

   c. Start all services.
2. Click the **Hosts** tab.
3. Optionally click the link of 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, RegionServer, 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 are 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. The Linux I/O scheduler controls when buffered writes are flushed to disk, based on time and quantity thresholds. It continually flushes buffered writes from multiple sources, not certain prioritized processes.

  Updates to this parameter are 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, you must restart the role for changes to 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 is unable to do so, it will kill the process. The kernel measures memory consumption in a manner that does not necessarily match what the `top` or `ps` report for RSS, so expect that this limit is a rough approximation.

  After updating this parameter, you must restart the role for changes to 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	**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	**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	**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 will not 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.

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 preempted and distributed to other pools. Otherwise, a pool receives a share of resources according to the pool's weight. Access control lists (ACLs) restrict who can submit work to dynamic resource pools and administer them.

A ***configuration set*** defines the allocation of resources across pools that can be active at a given time. For example, you can define "daytime" and "off hour" configuration sets, for which you specify different resource allocations during the daytime and for the remaining time of the week.

A ***scheduling rule*** defines when a configuration set is active. The configurations in the configuration set are propagated to the fair scheduler allocation file as required by the schedule. The updated files are stored in the YARN ResourceManager configuration directory `/var/run/cloudera-scm-agent/process/`*nn*`-yarn-RESOURCEMANAGER` on the host running the ResourceManager role. See Server and Client Configuration.

The resources available for sharing are subject to the allocations made for each service if static service pools (cgroups) are enforced. For example, if the static pool for YARN is 75% of the total cluster resources, resource pools will use only 75% of resources.

After you create or edit dynamic resource pool settings, **Refresh Dynamics Resource Pools** and **Discard Changes** buttons display. Click **Refresh Dynamics Resource Pools** to propagate the settings to the fair scheduler allocation file (by default, `fair-scheduler.xml`). The updated files are stored in the YARN ResourceManager configuration directory `/var/run/cloudera-scm-agent/process/`*nn*`-yarn-RESOURCEMANAGER` on the host running the ResourceManager role. See Server and Client Configuration.

For information on determining how allocated resources are used, see Cluster Utilization Reports on page 351.

Pool Hierarchy

YARN resource pools can be nested, with subpools restricted by the settings of their parent pool. This allows you to specify a set of pools whose resources are limited by the resources of a parent. Each subpool can have its own resource restrictions; if those restrictions fall within the configuration of the parent pool, the limits for the subpool take effect. If the limits for the subpool exceed those of the parent, the parent pool limits take precedence.

You create a parent pool either by configuring it as a parent or by creating a subpool under the pool. Once a pool is a parent, you cannot submit jobs to that pool; they must be submitted to a subpool.

Managing Dynamic Resource Pools

The main entry point for using dynamic resource pools with Cloudera Manager is the **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration** menu item.

Viewing Dynamic Resource Pool Configuration

Depending on which resource management scenario described in Cloudera Manager Resource Management on page 320 is in effect, the dynamic resource pool configuration overview displays the following tabs:

- **YARN** - Weight, Virtual Cores, Min and Max Memory, Max Running Apps, and Scheduling Policy

- **Impala Admission Control** - Max Memory, Max Running Queries, Max Queued Queries, Queue Timeout, and Default Query Memory Limit

To view dynamic resource pool configuration:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.

Creating a YARN Dynamic Resource Pool

There is always a resource pool named `root.default`. By default, all YARN applications run in this pool. You create additional pools when your workload includes identifiable groups of applications (such as from a particular application, or a particular group within your organization) that have their own requirements.

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** tab.
3. Click **Create Resource Pool**. The Create Resource Pool dialog box displays, showing the Configuration Sets tab.
4. Specify a name and resource limits for the pool:
   - In the **Resource Pool Name** field, specify a unique pool name containing only alphanumeric characters. If referencing a user or group name that contains a ".", replace the "." with "_dot_".
   - To make the pool a parent pool, select the **Parent Pool** checkbox.
   - Choose a configuration set. Specify a weight that indicates that pool's share of resources relative to other pools, minimum and maximum for virtual cores and memory, and a limit on the number of applications that can run simultaneously in the pool.
   - Click the **Scheduling Policy** tab and select a policy:
     - **Dominant Resource Fairness (DRF) (default)** - An extension of fair scheduling for more than one resource. DRF determines CPU and memory resource shares based on the availability of those resources and the job requirements.
     - **Fair (FAIR)** - Determines resource shares based on memory.
     - **First-In, First-Out (FIFO)** - Determines resource shares based on when a job was added.
   - If you have enabled Fair Scheduler preemption, click the **Preemption** tab and 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, follow the procedure in Enabling and Disabling Fair Scheduler Preemption on page 341.
   - 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. By default, anyone can submit, view all, and kill applications. To restrict these permissions, select **Allow these users and groups** and provide a comma-delimited list of users and groups in the Users and Groups fields respectively.
5. Click **Create**.
6. Click **Refresh Dynamic Resource Pools**.

Enabling and Disabling Dynamic Resource Pools for Impala

By default, admission control and dynamic resource pools for Impala are disabled. Only when both are enabled does the Impala Admission Control tab appear in the Dynamic Resource Pool Configuration tab. To enable and disable Impala dynamic resource pools, follow the procedure in Enabling and Disabling Impala Admission Control Using Cloudera Manager on page 351.

Creating an Impala Dynamic Resource Pool

There is always a resource pool designated as `root.default`. By default, all Impala queries run in this pool when the dynamic resource pool feature is enabled for Impala. You create additional pools when your workload includes identifiable groups of queries (such as from a particular application, or a particular group within your organization) that have their own requirements for concurrency, memory use, or service level agreement (SLA). Each pool has its own settings related to memory, number of queries, and timeout interval.

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **Impala Admission Control** tab.
3. Click **Create Resource Pool**.
4. Specify a name and resource limits for the pool:
   - In the **Resource Pool Name** field, type a unique name containing only alphanumeric characters.
   - Optionally click the **Submission Access Control** tab to specify which users and groups can submit queries. By default, anyone can submit queries. To restrict this permission, select the **Allow these users and groups** option and provide a comma-delimited list of users and groups in the Users and Groups fields respectively.
5. Click **Create**.
6. Click **Refresh Dynamic Resource Pools**.

Choosing Settings for an Impala Dynamic Resource Pool

Impala dynamic resource pools support the following settings:

Max Memory

The maximum amount of aggregate memory, cluster-wide, that can be used by all queries running concurrently in the pool. In conjunction with **Default Query Memory Limit** and the number of Impala hosts in the cluster, Impala determines the expected maximum memory used by all queries in the pool and holds back any further queries once the estimated upper limit is reached.

> **Note:** If you specify **Max Memory** for an Impala dynamic resource pool, you must also specify the **Default Query Memory Limit**. **Max Memory** relies on the **Default Query Memory Limit** to produce a reliable estimate of overall memory consumption for a query.

For example, consider the following scenario:

- The cluster is running `impalad` daemons on five DataNodes.
- A dynamic resource pool has **Max Memory** set to 100 GB.
- The **Default Query Memory Limit** for the pool is 10 GB. Therefore, any query running in this pool could use up to 50 GB of memory (default query memory limit * number of Impala nodes).
- The maximum number of queries that Impala executes concurrently within this dynamic resource pool is two, which is the most that could be accomodated within the 100 GB **Max Memory** cluster-wide limit.
- There is no memory penalty if queries use less memory than the **Default Query Memory Limit** per-host setting or the **Max Memory** cluster-wide limit. These values are only used to estimate how many queries can be run concurrently within the resource constraints for the pool.

Max Running Queries

The maximum number of queries that can run concurrently in this pool. The default value is unlimited. Any queries for this pool that exceed **Max Running Queries** are added to the admission control queue until other queries finish. You can use **Max Running Queries** in the early stages of resource management, when you do not have extensive data about query memory usage, to determine if the cluster performs better overall if throttling is applied to Impala queries.

For a workload with many small queries, you typically specify a high value for this setting, or leave the default setting of “unlimited”. For a workload with expensive queries, where some number of concurrent queries saturate the memory, I/O, CPU, or network capacity of the cluster, set the value low enough that the cluster resources are not overcommitted for Impala.

Once you have enabled memory-based admission control using other pool settings, you can still use **Max Running Queries** as a safeguard. If queries exceed either the total estimated memory or the maximum number of concurrent queries, they are added to the queue.

Max Queued Queries

The maximum number of queries that can be in the admission control queue for a pool at any one time. Additional queries cannot be added to the queue until other queries begin running, reducing the queue size. The default value is 200. Typically, this value does not need to be adjusted. If a large number of queries are queued, you can change other parameters such as the timeout interval, or you can increase the pool capacity.

Queue Timeout

The amount of time, in milliseconds, that a query waits in the admission control queue for this pool before being canceled. The default value is 60,000 (60 seconds).

It the following cases, **Queue Timeout** is not significant and you can specify a high value to avoid canceling queries unexpectedly:

- In a low-concurrency workload, where few or no queries are queued
- In an environment without a strict SLA, where it does not matter if queries occasionally take longer than usual because they are held in admission control

You might also need to increase the value to use Impala with some business intelligence tools that have their own timeout intervals for queries.

In a high-concurrency workload, especially for queries with a tight SLA, long wait times in admission control can cause a serious problem. For example, if a query needs to run in 10 seconds, and you have tuned it so that it runs in 8 seconds, it violates its SLA if it waits in the admission control queue longer than 2 seconds. In a case like this, set a low timeout value and monitor how many queries are cancelled because of timeouts. This technique helps you to discover capacity, tuning, and scaling problems early, and helps avoid wasting resources by running expensive queries that have already missed their SLA.

If you identify some queries that can have a high timeout value, and others that benefit from a low timeout value, you can create separate pools with different values for this setting.

Default Query Memory Limit

The equivalent of setting the `MEM_LIMIT` query option for each query that runs in this pool. This value represents the maximum memory the query can use on each host. If the query exceeds that memory on a host, it activates spill-to-disk, and could be canceled if available memory is too low. Impala multiplies this default memory limit value by the number of Impala hosts in the cluster to estimate how many queries fit within the total RAM represented by **Max Memory**, which represents a cluster-wide limit.

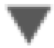
Cloning a Resource Pool

To create a new pool with the same properties as an existing pool, you can clone a pool:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.
3. Click at the right of a resource pool row and select **Clone**.
4. Specify a name for the pool and edit pool properties as desired.
5. Click **Create**.
6. Click **Refresh Dynamic Resource Pools**.

Deleting a Resource Pool

To delete a pool:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.
3. Click  at the right of a resource pool row and select **Delete**.
4. Click **OK**.
5. Click **Refresh Dynamic Resource Pools**.

Editing Dynamic Resource Pools

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.
3. Click **Edit** at the right of a resource pool row. Edit the properties.
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. By default, anyone can submit, view all, and kill applications. To restrict these permissions, select **Allow these users and groups** and provide a comma-delimited list of users and groups in the Users and Groups fields respectively.
5. Click **Save**.
6. Click **Refresh Dynamic Resource Pools**.

YARN Pool Status and Configuration Options

Viewing Dynamic Resource Pool Status

Do one of the following:

- **Cluster** menu
  1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.
  2. Click **View Dynamic Resource Pool Status**.
- YARN service
  1. Go to the YARN service.
  2. Click the **Resource Pools** tab.

See Monitoring Dynamic Resource Pools for more information.

Configuring Default YARN Fair Scheduler Properties

For information on the default properties, see Configuring the Fair Scheduler on page 338.

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.
2. Click the **YARN** tab.
3. Click the **Default Settings** button.
4. Specify the default scheduling policy, maximum applications, and preemption timeout properties.
5. Click **Save**.
6. Click **Refresh Dynamic Resource Pools**.

Creating a Resource Subpool

YARN resource pools can form a nested hierarchy. To manually create a subpool:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.

2. Click ▼ at the right of a resource pool row and select **Create Subpool**. Configure subpool properties.
3. Click **Create**.
4. Click **Refresh Dynamic Resource Pools**.

Setting YARN 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:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.
2. Click the **User Limits** tab. The table displays a list of users and the maximum number of jobs each user can submit.
3. Click **Add User Limit**.
4. Specify a username. containing only alphanumeric characters. If referencing a user or group name that contains a ".", replace the "." with "_dot_".
5. Specify the maximum number of running applications.
6. Click **Create**.
7. Click **Refresh Dynamic Resource Pools**.

Configuring ACLs

To configure which users and groups can submit and kill applications in any resource pool:

1. Enable ACLs.
2. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.
3. Click the **Access Control Settings** button.
4. In the **Admin ACL** property, specify which users and groups can submit and kill applications.

> **Note:** Group names in YARN Resource Manager ACLs are case sensitive. Consequently, if you specify an uppercase group name in the ACL, it will not match the group name resolved from the Active Directory because the Active Directory group name is resolved in lowercase.

5. Click **Save Changes** to commit the changes.
6. Return to the Home page by clicking the Cloudera Manager logo.
7. Click ⏻ to invoke the cluster restart wizard.
8. Click **Restart Stale Services**.
9. Click **Restart Now**.
10. Click **Finish**.

Enabling ACLs

To specify whether ACLs are checked:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. The **YARN** > **Resource Pools** tab displays.
2. Click the **Access Control Settings** button.
3. In the **Enable ResourceManager ACLs** property, select the YARN service.
4. Click **Save Changes** to commit the changes.
5. Return to the Home page by clicking the Cloudera Manager logo.
6. Click ⏻ to invoke the cluster restart wizard.
7. Click **Restart Stale Services**.
8. Click **Restart Now**.
9. Click **Finish**.

Defining Resource Allocations with Configuration Sets

A ***configuration set*** defines the allocation of resources across pools that can be active at a given time. For example, you can define "daytime" and "off hour" configuration sets, for which you specify different resource allocations during the daytime and for the remaining time of the week.

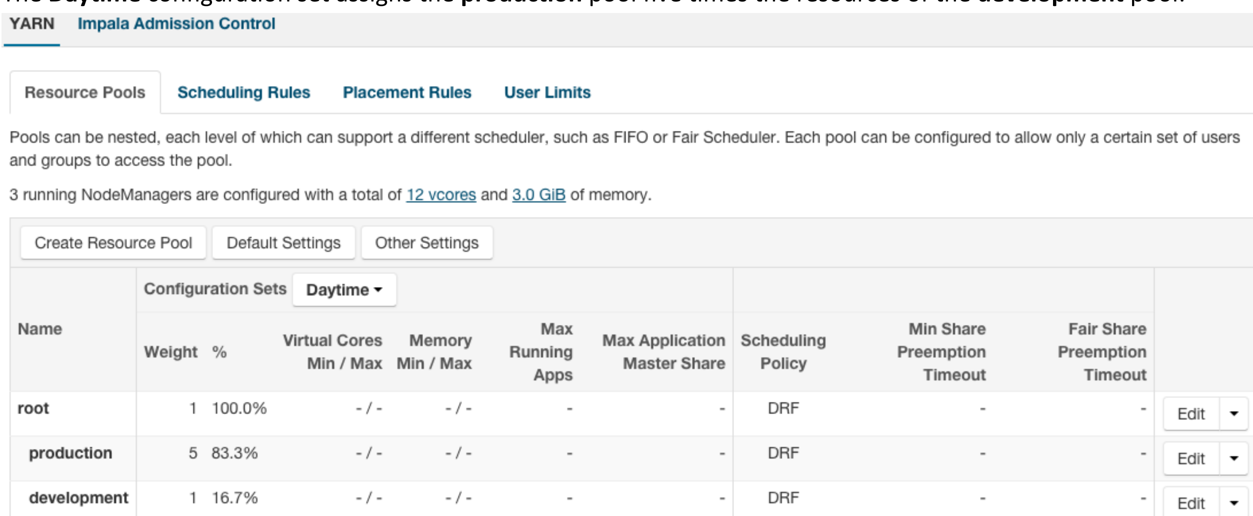
You define configuration set activity by creating [scheduling rules](). Once you have created a configuration set, you can configure its properties, such as weight, minimum and maximum memory and virtual cores, and maximum running applications.

Specifying Configuration Set Properties

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service [enabled for dynamic resource pools](), the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **Resource Pools** tab.
3. For each resource pool, click **Edit**.
    a. Select a configuration set name.
    b. Edit the configuration set properties and click **Save**.
4. Click **Refresh Dynamic Resource Pools**.

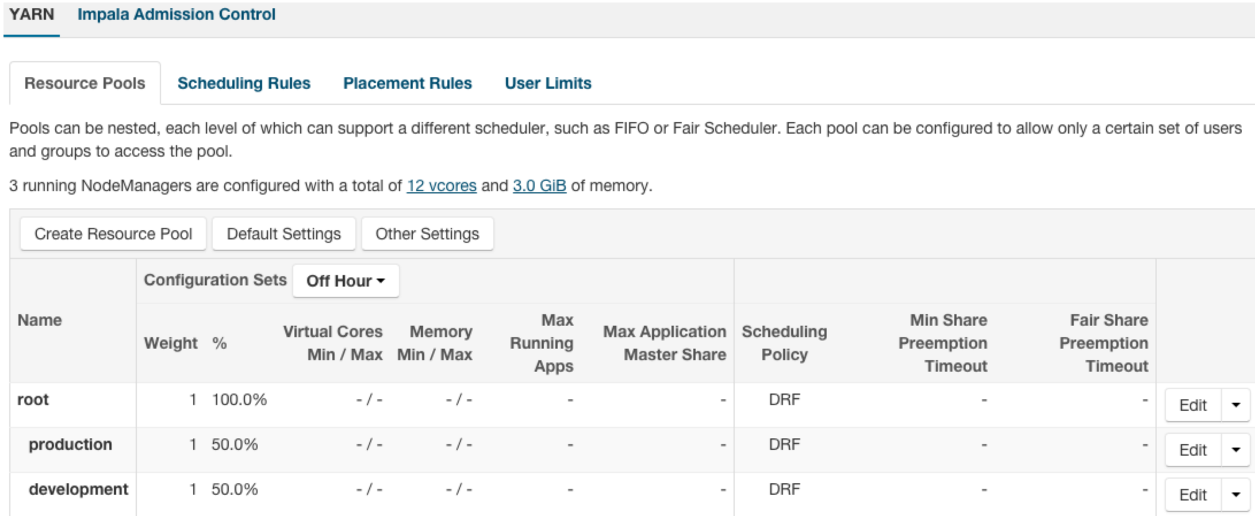
Example Configuration Sets

The **Daytime** configuration set assigns the **production** pool five times the resources of the **development** pool:



Name	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Max Application Master Share	Scheduling Policy	Min Share Preemption Timeout	Fair Share Preemption Timeout	
root	1	100.0%	- / -	- / -	-	-	DRF	-	-	Edit
production	5	83.3%	- / -	- / -	-	-	DRF	-	-	Edit
development	1	16.7%	- / -	- / -	-	-	DRF	-	-	Edit

The **Off Hour** configuration set assigns the **production** and **development** pools an equal amount of resources:



YARN | Impala Admission Control

Resource Pools | Scheduling Rules | Placement Rules | User Limits

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.

3 running NodeManagers are configured with a total of 12 vcores and 3.0 GiB of memory.

Create Resource Pool | Default Settings | Other Settings

Configuration Sets: Off Hour

Name	Weight	%	Virtual Cores Min / Max	Memory Min / Max	Max Running Apps	Max Application Master Share	Scheduling Policy	Min Share Preemption Timeout	Fair Share Preemption Timeout	
root	1	100.0%	- / -	- / -	-	-	DRF	-	-	Edit
production	1	50.0%	- / -	- / -	-	-	DRF	-	-	Edit
development	1	50.0%	- / -	- / -	-	-	DRF	-	-	Edit

See example scheduling rules for these configuration sets.

Viewing the Properties of a Configuration Set

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. In the **Configuration Sets** drop-down list, select a configuration set. The properties of each pool for that configuration set display.

Scheduling Configuration Sets

A ***scheduling rule*** defines when a configuration set is active. The configurations in the configuration set are propagated to the fair scheduler allocation file as required by the schedule. The updated files are stored in the YARN ResourceManager configuration directory `/var/run/cloudera-scm-agent/process/nn-yarn-RESOURCEMANAGER` on the host running the ResourceManager role. See Server and Client Configuration.

Example Scheduling Rules

Consider the example Daytime and Off Hour configuration sets. To specify that the **Daytime** configuration set is active every weekday from 8:00 a.m. to 5:00 p.m. and the **Off Hour** configuration set is active all other times (evenings and weekends), define the following rules:

	Scheduling Rule	Configuration Set
1	Repeats weekly on Monday, Tuesday, Wednesday, Thursday, Friday from 8:00 AM to 5:00 PM (PST), starting 03/08/2016.	**Daytime**
2	Repeats weekly on Monday, Tuesday, Wednesday, Thursday, Friday from 5:00 PM to 8:00 AM (PST), starting 03/08/2016.	**Off Hour**
3	Repeats weekly on Sunday, Saturday from 12:00 AM to 12:00 AM (PST), starting 03/08/2016.	**Off Hour**

Adding a Scheduling Rule

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **Scheduling Rules** tab.
3. Click **Create Scheduling Rule**.
4. In the Configuration Set field, choose the configuration set to which the rule applies. Select **Create New** or **Use Existing**.
5. If you create a new configuration set, type a name in the Name field.

   If you use an existing configuration set, select one from the drop-down list.

6. Configure the rule to repeat or not:

   - To repeat the rule, keep the **Repeat** field selected and specify the repeat frequency. If the frequency is weekly, specify the repeat day or days.
   - If the rule does not repeat, clear the **Repeat** field, 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.

7.
8. Click **Create**.
9. Click **Refresh Dynamic Resource Pools**.

Reordering Scheduling Rules

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **Scheduling Rules** tab.
3. Click **Reorder Scheduling Rules**.
4. Click **Move Up** or **Move Down** in a rule row.
5. Click **Save**.
6. Click **Refresh Dynamic Resource Pools**.

Editing a Scheduling Rule

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click **Scheduling Rules**.
3. Click **Edit** at the right of a rule.
4. Edit the rule.
5. Click **Save**.
6. Click **Refresh Dynamic Resource Pools**.

Deleting a Scheduling Rule

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click **Scheduling Rules**.
3. Click  at the right of a rule and select **Delete**.
4. Click **OK**.
5. Click **Refresh Dynamic Resource Pools**.

Assigning Applications and Queries to Resource Pools

You can specify at run time that an application or query should run in a named resource pool. To specify a pool at run time for a YARN application, provide the pool name in the `mapreduce.job.queuename` property. To specify a pool at run time for an Impala query, provide the pool name in the `REQUEST_POOL` option.

Cloudera Manager allows you to specify a set of ordered rules for assigning applications and queries to pools. You can also specify default pool settings directly in the YARN fair scheduler configuration.

Some rules allow to you specify that the pool be created in the dynamic resource pool configuration if it does not already exist. Allowing pools to be created is optional. If a rule is satisfied and you do not create a pool, YARN runs the job "ad hoc" in a pool to which resources are not assigned or managed.

If *no rule is satisfied* when the application or query runs, the YARN application or Impala query is rejected.

Placement Rule Ordering and Evaluation

Pool placement rules are evaluated in the order in which they appear in the placement rule list. When a job is submitted, the rules are evaluated, and the first matching rule is used to determine the pool in which the job is run.

If a rule is always satisfied, subsequent rules are not evaluated. Rules that are never evaluated appear in struck-though gray text. For an example see Example Placement Rules on page 337.

By default, pool placement rules are ordered in reverse order of when they were added; the last rule added appears first. You can easily reorder rules.

Creating Pool Placement Rules

Placement rules determine the pools to which applications and queries are assigned. At installation, Cloudera Manager provides a set of default rules and rule ordering.

To create a placement rule:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.
3. Click the **Placement Rules** tab.
4. Click **Create Placement Rule**. In the **Type** drop-down list, select a rule that specifies the name of pool and its position in the pool hierarchy:

   - **YARN**
     - **specified at run time** - Use `root.[pool name]`, where `pool name` is the name of the pool specified at run time.
     - **root.users.[username]** - Use the parent pool `root.users` in a pool named by the user submitting the application. The `root.users` parent pool and this rule are created by default. However, on upgrading from Cloudera Manager 5.7, neither the pool or placement rule is added.
     - **root.default** - Use the `root.default` pool.
     - **root.[pool name]** - Use `root.pool name`, where `pool name` is the name you specify in the **Pool Name** field that displays after you select the rule.
     - **root.[primary group]** - Use the pool that matches the primary group of the user submitting the application.
     - **root.[secondary group]** - Use the pool that matches one of the secondary groups of the user that submitted the application.
     - **root.[username]** - Use the pool that matches the name of the user that submitted the application.
     - **root.[primary group].[username]** - Use the parent pool that matches the primary group of the user that submitted the application and then a subpool that matches the username.
     - **root.[secondary group].[username]** - Use the parent pool that matches one of the secondary groups of the user that submitted the application and then a subpool that matches the username.

   - **Impala**
     - **specified at run time** - Use the `REQUEST_POOL` query option. For example, `SET REQUEST_POOL=root.[pool name]`.
     - **root.[pool name]** - Use `root.pool name`, where `pool name` is the name you specify in the **Pool Name** field that displays after you select the rule.
     - **root.[username]** - Use the pool that matches the name of the user that submitted the query. This is not recommended.
     - **root.[primary group]** - Use the pool that matches the primary group of the user that submitted the query.

- **root.[secondary group]** - Use the pool that matches one of the secondary groups of the user that submitted the query.
- **root.default** - Use the `root.default` pool.

> **Note:** Currently, it's not possible to map a username to a resource pool with different name. For example, you cannot map the `user1` user to the `root.pool1` resource pool.

For example, consider the following three Impala users.

- *username*: test1, *secondarygroupname*: testgroup
- *username*: test2, *secondarygroupname*: testgroup
- *username*: test3, *secondarygroupname*: testgroup

This cluster has 2 dynamic pools: `root.default` and `root.testgroup`

This cluster has 3 placement rules listed in the following order.

1. `root.[username]`
2. `root.[primary group]`
3. `root.[secondary group]`

With the above placement rule, query submitted by all three users will be mapped to the `root.testgroup` resource pool.

For more information about these rules, see the description of the `queuePlacementPolicy` element in Allocation File Format.

5. **(YARN only)** To indicate that the pool should be created if it does not exist when the application runs, check the **Create pool if it does not exist** checkbox.
6. Click **Create**. The rule is added to the top of the placement rule list and becomes the first rule evaluated.
7. Click **Refresh Dynamic Resource Pools**.

Default Placement Rules and Order

The default placement rules and order for YARN are:

1. Use the pool specified at run time and create the pool if it does not exist.
2. Use the pool root.users.[username].
3. Use the pool root.default.

Also see Example Placement Rules on page 337.

The default rules and order for Impala are:

1. Use the pool specified at run time, only if the pool exists.
2. Use the pool root.default.

Reordering Pool Placement Rules

To change the order in which pool placement rules are evaluated:

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays. If the cluster has an Impala service enabled for dynamic resource pools, the **Impala Admission Control** > **Resource Pools** tab displays.
2. Click the **YARN** or **Impala Admission Control** tab.
3. Click the **Placement Rules** tab.
4. Click **Reorder Placement Rules**.
5. Click **Move Up** or **Move Down** in a rule row.

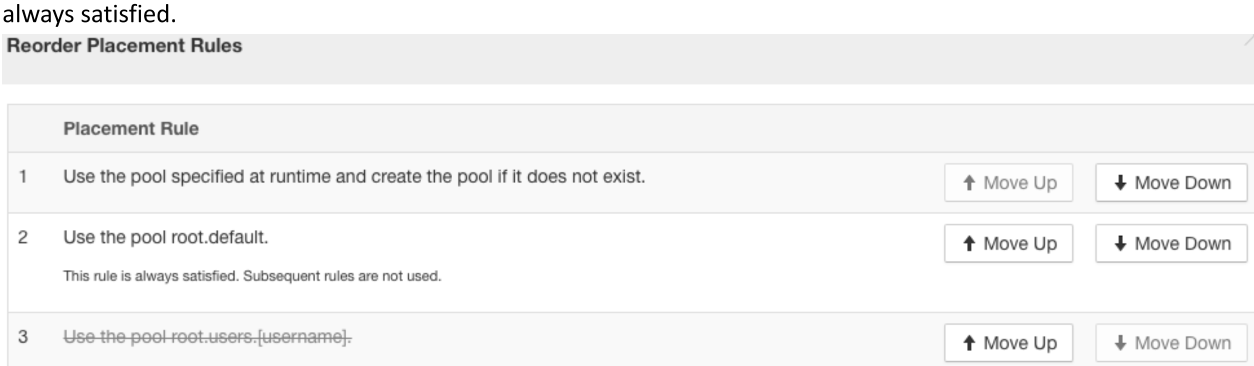
6. Click **Save**.
7. Click **Refresh Dynamic Resource Pools**.

Example Placement Rules

The following figures show the default pool placement rule setting for YARN:

If a pool is specified at run time, that pool is used for the job and the pool is created if it did not exist. If no pool is specified at run time, a pool named according to the user submitting the job within the `root.users` parent pool is used. If that pool cannot be used (for example, because the `root.users` pool is a leaf pool), pool `root.default` is used.

If you move rule 2 down (which specifies to run the job in a pool named after the user running the job nested within the parent pool `root.users`), rule 2 becomes disabled because the previous rule (Use the pool `root.default`) is always satisfied.



YARN (MRv2) and MapReduce (MRv1) Schedulers

A ***scheduler*** determines which jobs run, where and when they run, and the resources allocated to the jobs. The YARN (MRv2) and MapReduce (MRv1) 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. When configuring the scheduling policy of a pool, Domain Resource Fairness (DRF) is a type of fair scheduler.
  - 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 YARN and MapReduce are:

- **YARN** - Cloudera Manager, CDH 5, and CDH 4 set the default to Fair Scheduler. Cloudera recommends Fair Scheduler. 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 on 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, and network.

The YARN scheduler has a pluggable policy, which is responsible for partitioning cluster resources among the various queues, applications, and so on.

If you are running CDH 5, you can manually configure the scheduler type. If you choose the Fair Scheduler, see Configuring the Fair Scheduler on page 338 for information on how to manually configure it. Alternatively you can use Cloudera Manager dynamic allocation to manage scheduler configuration.

- **MapReduce** - Cloudera Manager, CDH 5, and CDH 4 set the default scheduler to FIFO for backward compatibility, however Cloudera recommends Fair Scheduler. Capacity Scheduler is also available.

  If you are running CDH 4, you can specify how MapReduce jobs share resources by configuring the MapReduce scheduler.

Configuring the Fair Scheduler

The Fair Scheduler is the Cloudera recommended scheduler option. The Fair Scheduler controls how resources are allocated to ***pool***s (or ***queue***s) and how jobs are assigned to pools. Jobs can also be explicitly submitted to pools; to submit an job to a specific pool, you specify the `mapreduce.job.queuename` property.

Pools have policies for preempting running jobs, so that when there is contention for resources, jobs that have high priority or have been waiting a long time to run are allowed to run.

Fair Scheduler configuration is maintained in two files: `yarn-site.xml` and `fair-scheduler.xml`. Detailed information on the available properties is available at Fair Scheduler Configuration. When you change the contents of `yarn-site.xml`, you must restart the YARN service.

In Cloudera Manager the Dynamic Resource Pools Configuration screen provides an enhanced interface for configuring the Fair Scheduler. In addition to allowing you to configure resource allocation properties, you can define schedules for changing the values of the properties. Cloudera Manager automatically updates Fair Scheduler configuration files according to the schedule.

Table 22: yarn-site.xml Properties

Property	Description
yarn.scheduler.fair.allow-undeclared-pools	When set to **true**, the Fair Scheduler uses the username as the default pool name, in the event that a pool name is not specified. When set to **false**, all applications are run in a shared pool, called **default**. Default: true.
yarn.scheduler.fair.user-as-default-queue	When set to **true**, pools specified in applications but not explicitly configured, are created at runtime with default settings. When set to **false**, applications specifying pools not explicitly configured run in a pool named **default**. This setting applies when an application explicitly specifies a pool and when the application runs in a pool named with the username associated with the application. Default: true.
yarn.scheduler.fair.preemption	When enabled, under certain conditions, the Fair Scheduler preempts applications in other pools. Preemption guarantees that production applications are not starved while also allowing the cluster to be used for experimental and research applications. To minimize wasted computation, the Fair Scheduler preempts the most recently launched applications. Default: false.

Property	Description
yarn.scheduler.fair.preemption.cluster-utilization-threshold	The cluster utilization threshold above which preemption is triggered. If the cluster utilization is under this threshold, preemption is not triggered even if there are starved queues. The utilization is computed as the maximum ratio of usage to capacity among all resources. Default: .8.

For example:

```
...
  <property>
    <name>yarn.scheduler.fair.allow-undeclared-pools</name>
    <value>true</value>
  </property>
  <property>
    <name>yarn.scheduler.fair.user-as-default-queue</name>
    <value>true</value>
  </property>
  <property>
    <name>yarn.scheduler.fair.preemption</name>
    <value>true</value>
  </property>
  <property>
    <name>yarn.scheduler.fair.preemption.cluster-utilization-threshold</name>
    <value>0.8</value>
  </property>
...
```

Figure 10: yarn-site.xml Example

Table 23: fair-scheduler.xml Properties

Element	Subelement	Description
queuePlacementPolicy		Policy for assigning jobs to resource pools. In Cloudera Manager this property is configured using placement rules.
userMaxAppsDefault		Default running app limit for a user whose limit is not otherwise specified. In Cloudera Manager this property is configured using user limits.
queueMaxAppsDefault		Default running app limit for pools; overridden by the equivalent element in a pool.
queueMaxAMShareDefault		Default ApplicationMaster resource limit for the pool; overridden by the equivalent element in a pool.
defaultFairSharePreemptionThreshold		Fair share preemption threshold for pools; overridden by the equivalent element in a pool. The threshold value is between 0 and 1. If set to x and the fair share of the pool is F, resources are preempted from other pools if the allocation is less than (x * F).
defaultFairSharePreemptionTimeout		Default number of seconds a resource pool is under its fair share before it will preempt containers to take resources from other resource pools; overridden by the equivalent element in a pool. If this parameter is not set, and fairSharePreemptionTimeout is not set for a given queue or its ancestor queues, pre-emption by this queue will never occur, even if pre-emption is enabled. Default timeout is 2^64 milliseconds.
defaultMinSharePreemptionTimeout		Default number of seconds a resource pool is under its minimum share before it will preempt containers to take resources from other resource pools; overridden by the equivalent element in a pool.

Element	Subelement	Description
defaultQueueSchedulingPolicy		Default scheduling policy for pools; overridden by the equivalent element in a pool. Default: drf.
queue		Name of a dynamic resource pool.
	weight	Weight given to the resource pool when determining how to allocate resources relative to other resource pools. In Cloudera Manager this property is configured using configuration sets.
	schedulingPolicy	Policy to determine how resources are allocated to the resource pool: fair, fifo, or drf.
	aclSubmitApps	Users and groups that can submit jobs to the pool.
	aclAdministerApps	Users and groups that can administer the pool.
	minResources, maxResources	Minimum and maximum share of resources that can allocated to the resource pool in the form *X* mb, *Y* vcores. Values computed by the weight settings are limited by (or constrained by) the minimum and maximum values.
	maxAMShare	Fraction of the resource pool's fair share that can be used to run ApplicationMasters. For example, if set to 1.0, then ApplicationMasters in the pool can take up to 100% of both the memory and CPU fair share. The value of -1.0 disables this feature, and the ApplicationMaster share is not checked. The default value is 0.5.
	maxRunningApps	See default elements.
	fairSharePreemptionThreshold	See default elements.
	fairSharePreemptionTimeout	See default elements.
	minSharePreemptionTimeout	See default elements.

For example:

```
<allocations>
    <queue name="root">
        <weight>1.0</weight>
        <schedulingPolicy>drf</schedulingPolicy>
        <aclSubmitApps> </aclSubmitApps>
        <aclAdministerApps>*</aclAdministerApps>
        <queue name="production">
            <minResources>1024 mb, 10 vcores</minResources>
            <maxResources>5120 mb, 20 vcores</maxResources>
            <weight>4.0</weight>
            <schedulingPolicy>drf</schedulingPolicy>
            <aclSubmitApps>*</aclSubmitApps>
            <aclAdministerApps>*</aclAdministerApps>
        </queue>
        <queue name="development">
            <weight>1.0</weight>
            <schedulingPolicy>drf</schedulingPolicy>
            <aclSubmitApps>*</aclSubmitApps>
            <aclAdministerApps>*</aclAdministerApps>
        </queue>
    </queue>
    <defaultQueueSchedulingPolicy>drf</defaultQueueSchedulingPolicy>
    <queuePlacementPolicy>
        <rule name="specified" create="true"/>
        <rule name="user" create="true"/>
```

```
    </queuePlacementPolicy>
</allocations>
```

Figure 11: fair-scheduler.xml

Dynamic resource pools allow you to configure scheduler properties. See Configuring Default YARN Fair Scheduler Properties on page 330.

Enabling and Disabling Fair Scheduler Preemption

You can enable the Fair Scheduler to preempt applications in other pools if a pool's fair or minimum share is not met for some period of time. When you create a pool, you can specify whether preemption is allowed and whether a specific pool can be preempted. Fair scheduler preemption is controlled by several properties. For more information, see Configuring the Fair Scheduler on page 338.

Enabling and Disabling Preemption Using Cloudera Manager

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

Enabling Preemption

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays.
2. Click **Default Settings**.
3. Click the **Enable Fair Scheduler Preemption** link.
4. Select the **ResourceManager Default Group** checkbox.
5. Click **Save Changes** to commit the changes.
6. Return to the Home page by clicking the Cloudera Manager logo.
7. Click ⏻ to invoke the cluster restart wizard.
8. Click **Restart Stale Services**.
9. Click **Restart Now**.
10. Click **Finish**.

Disabling Preemption

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays.
2. Click the **Access Control Settings** button.
3. Clear the **Enable Fair Scheduler Preemption** checkbox.
4. Click **Save Changes** to commit the changes.
5. Return to the Home page by clicking the Cloudera Manager logo.
6. Click ⏻ to invoke the cluster restart wizard.
7. Click **Restart Stale Services**.
8. Click **Restart Now**.
9. Click **Finish**.

Enabling and Disabling Preemption for a Pool

1. Select **Clusters** > ***Cluster name*** > **Dynamic Resource Pool Configuration**. If the cluster has a YARN service, the **YARN** > **Resource Pools** tab displays.
2. In a pool row, click **Edit**.
3. Click the **Preemption** tab.
4. Select or clear **Preemption**.
5. Click **Save**.
6. Click **Refresh Dynamic Resource Pools**.

Enabling and Disabling Preemption Using the Command Line

Important:

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

To enable and disable preemption, set `yarn.scheduler.fair.preemption` in `yarn-site.xml`:

```
<property>
    <name>yarn.scheduler.fair.preemption</name>
    <value>true</value>
</property>
```

To enable or disable a specific pool from being preempted, set the `allowPreemptionFrom` property in `fair-scheduler.xml`. The following example disables preemption from the `important` pool:

```
<queue name="important">
    <weight>1.0</weight>
    <allowPreemptionFrom>false</allowPreemptionFrom>
    <schedulingPolicy>drf</schedulingPolicy>
    <aclSubmitApps>*</aclSubmitApps>
    <aclAdministerApps>*</aclAdministerApps>
</queue>
```

Resource Management for Impala

Note:

The use of the Llama component for integrated resource management within YARN is no longer supported with CDH 5.5 / Impala 2.3 and higher.

For clusters running Impala alongside other data management components, you define static service pools to define the resources available to Impala and other components. Then within the area allocated for Impala, you can create dynamic service pools, each with its own settings for the Impala admission control feature.

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.

How Resource Limits Are Enforced

- If Cloudera Manager Static Partitioning is used, it creates a cgroup in which Impala runs. This cgroup limits CPU, network, and IO according to the static partitioning policy.
- Limits on memory usage are enforced by Impala's process memory limit (the `MEM_LIMIT` query option setting). The admission control feature checks this setting to decide how many queries can be safely run at the same time. Then the Impala daemon enforces the limit by activating the spill-to-disk mechanism when necessary, or cancelling a query altogether if the limit is exceeded at runtime.

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:

- EXPLAIN_LEVEL Query Option
- MEM_LIMIT Query Option

Limitations of Resource Management for Impala

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.

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.

Queries, DML statements, and some DDL statements, including `CREATE TABLE AS SELECT` and `COMPUTE STATS` are affected by admission control.

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.

> **Note:**
>
> The use of the Llama component for integrated resource management within YARN is no longer supported with CDH 5.5 / Impala 2.3 and higher.
>
> For clusters running Impala alongside other data management components, you define static service pools to define the resources available to Impala and other components. Then within the area allocated for Impala, you can create dynamic service pools, each with its own settings for the Impala admission control feature.

Overview of Impala Admission Control

On a busy CDH cluster, you might find there is an optimal number of Impala queries that run concurrently. For example, when the I/O capacity is fully utilized by I/O-intensive queries, you might not find any throughput benefit in running more concurrent queries. By allowing some queries to run at full speed while others wait, rather than having all queries contend for resources and run slowly, admission control can result in higher overall throughput.

For another example, consider a memory-bound workload such as many large joins or aggregation queries. Each such query could briefly use many gigabytes of memory to process intermediate results. Because Impala by default cancels queries that exceed the specified memory limit, running multiple large-scale queries at once might require re-running some queries that are cancelled. In this case, admission control improves the reliability and stability of the overall workload by only allowing as many concurrent queries as the overall memory of the cluster can accommodate.

The admission control feature lets you set an 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.

In CDH 5.7 / Impala 2.5 and higher, you can specify these limits and thresholds for each pool rather than globally. That way, you can balance the resource usage and throughput between steady well-defined workloads, rare resource-intensive queries, and ad hoc exploratory queries.

For details on the internal workings of admission control, see How Impala Schedules and Enforces Limits on Concurrent Queries on page 345.

Concurrent Queries and Admission Control

One way to limit resource usage through admission control is to set an upper limit on the number of concurrent queries. This is the initial technique you might use when you do not have extensive information about memory usage for your workload. This setting can be specified separately for each dynamic resource pool.

You can combine this setting with the memory-based approach described in Memory Limits and Admission Control on page 344. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

See Managing Dynamic Resource Pools on page 326 for information about all these dynamic resource pool settings, how to use them together, and how to divide different parts of your workload among different pools.

Memory Limits and Admission Control

Each dynamic resource pool can have an upper limit on the cluster-wide memory used by queries executing in that pool. This is the technique to use once you have a stable workload with well-understood memory requirements.

Always specify the **Default Query Memory Limit** for the expected maximum amount of RAM that a query might require on each host, which is equivalent to setting the `MEM_LIMIT` query option for every query run in that pool. That value affects the execution of each query, preventing it from overallocating memory on each host, and potentially activating the spill-to-disk mechanism or cancelling the query when necessary.

Optionally, specify the **Max Memory** setting, a cluster-wide limit that determines how many queries can be safely run concurrently, based on the upper memory limit per host multiplied by the number of Impala nodes in the cluster.

For example, consider the following scenario:

- The cluster is running `impalad` daemons on five DataNodes.
- A dynamic resource pool has **Max Memory** set to 100 GB.
- The **Default Query Memory Limit** for the pool is 10 GB. Therefore, any query running in this pool could use up to 50 GB of memory (default query memory limit * number of Impala nodes).
- The maximum number of queries that Impala executes concurrently within this dynamic resource pool is two, which is the most that could be accomodated within the 100 GB **Max Memory** cluster-wide limit.
- There is no memory penalty if queries use less memory than the **Default Query Memory Limit** per-host setting or the **Max Memory** cluster-wide limit. These values are only used to estimate how many queries can be run concurrently within the resource constraints for the pool.

> **Note:** If you specify **Max Memory** for an Impala dynamic resource pool, you must also specify the **Default Query Memory Limit**. **Max Memory** relies on the **Default Query Memory Limit** to produce a reliable estimate of overall memory consumption for a query.

You can combine the memory-based settings with the upper limit on concurrent queries described in Concurrent Queries and Admission Control on page 344. If either the maximum number of or the expected memory usage of the concurrent queries is exceeded, subsequent queries are queued until the concurrent workload falls below the threshold again.

See Managing Dynamic Resource Pools on page 326 for information about all these dynamic resource pool settings, how to use them together, and how to divide different parts of your workload among different pools.

How Impala Admission Control Relates to Other Resource Management Tools

The admission control feature is similar in some ways to the Cloudera Manager static partitioning feature, as well as the YARN resource management framework. These features can be used separately or together. This section describes some similarities and differences, to help you decide which combination of resource management features to use for Impala.

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 does not interact with other Hadoop workloads such as MapReduce jobs, you might use YARN with static service pools on CDH 5 clusters where resources are shared between Impala and other Hadoop components. This configuration is recommended when using Impala in a ***multitenant*** cluster. Devote a percentage of cluster resources to Impala, and 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 work for other components within the cluster. In this scenario, Impala's resources are not managed by YARN.

The Impala admission control feature uses the same configuration mechanism as the YARN resource manager to map users to pools and authenticate them.

Although the Impala admission control feature uses a `fair-scheduler.xml` configuration file behind the scenes, this file does not depend on which scheduler is used for YARN. You still use this file, and Cloudera Manager can generate it for you, even when YARN is using the capacity scheduler.

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 across many coordinators. There could be times when the more queries were queued (in aggregate across the cluster) than the specified limit, or when number of admitted queries exceeds the expected number. 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 configuring memory resources, to leave some headroom in case more queries are admitted than expected, without running out of memory and being cancelled as a result.

To avoid a large backlog of queued requests, you can set an upper limit on the size of the queue for queries that are queued. When the number of queued queries exceeds this limit, further queries are cancelled rather than being queued. You can also configure a timeout period per pool, 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.

In Impala 2.0 and higher, you can submit a SQL SET statement from the client application to change the `REQUEST_POOL` query option. This option lets you submit queries to different resource pools, as described in REQUEST_POOL Query Option.

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.

Admission control has the following limitations or special behavior when used with JDBC or ODBC applications:

- The other resource-related query options, `RESERVATION_REQUEST_TIMEOUT` and `V_CPU_CORES`, are no longer used. Those query options only applied to using Impala with Llama, which is no longer supported.

SQL and Schema Considerations for Admission Control

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. Before setting up admission control, tune your Impala queries to ensure that the query plans are efficient and the memory estimates are accurate. Understanding the nature of your workload, and which queries are the most resource-intensive, helps you to plan how to divide the queries into different pools and decide what limits to define for each pool.

For large tables, especially those involved in join queries, keep their statistics up to date after loading substantial amounts of new data or adding new partitions. Use the `COMPUTE STATS` statement for unpartitioned tables, and `COMPUTE INCREMENTAL STATS` for partitioned tables.

When you use dynamic resource pools with a **Max Memory** setting enabled, you typically override the memory estimates that Impala makes based on the statistics from the `COMPUTE STATS` statement. You either set the `MEM_LIMIT` query option within a particular session to set an upper memory limit for queries within that session, or a default `MEM_LIMIT` setting for all queries processed by the `impalad` instance, or a default `MEM_LIMIT` setting for all queries assigned to a particular dynamic resource pool. By designating a consistent memory limit for a set of similar queries that use the same resource pool, you avoid unnecessary query queuing or out-of-memory conditions that can arise during high-concurrency workloads when memory estimates for some queries are inaccurate.

Follow other steps from Tuning Impala for Performance to tune your queries.

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). Cloudera recommends configuring the settings through the Cloudera Manager user interface.

> **Important:** Although the following options are still present in the Cloudera Manager interface under the **Admission Control** configuration settings dialog, Cloudera recommends you not use them in CDH 5.7 / Impala 2.5 and higher. These settings only apply if you enable admission control but leave dynamic resource pools disabled. In CDH 5.7 / Impala 2.5 and higher, prefer to set up dynamic resource pools and customize the settings for each pool, as described in Creating an Impala Dynamic Resource Pool on page 328 and Editing Dynamic Resource Pools on page 330.

Impala Service Flags for Admission Control (Advanced)

The following Impala configuration options let you adjust the settings of the admission control feature. When supplying the options on the `impalad` command line, prepend the option name with `--`.

`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`

`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: -1, meaning unlimited (prior to CDH 5.7 / Impala 2.5, the default was 200)

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: unlimited

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: `false`

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 configuration file used by admission control (`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 few of the settings that can go in this file, as described below.

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.

Configuring Admission Control Using the Command Line

To configure admission control, use a combination of startup options for the Impala daemon and edit or create 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 `impalad` 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
impala.admission-control.pool-default-query-options.queue_name
impala.admission-control.pool-queue-timeout-ms.queue_name
```

The `impala.admission-control.pool-queue-timeout-ms` setting specifies the timeout value for this pool, in milliseconds. The`impala.admission-control.pool-default-query-options` settings designates the default query options for all queries that run in this pool. Its argument value is a comma-delimited string of 'key=value' pairs, for example,`'key1=val1,key2=val2'`. For example, this is where you might set a default memory limit for all queries in the pool, using an argument such as `MEM_LIMIT=5G`.

The `impala.admission-control.*` configuration settings are available in CDH 5.7 and higher.

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 326.

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 `root.default`, `root.development`, and `root.production`. These sample files are stripped down: in a real deployment they might contain other settings for use with various aspects of the YARN component. 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>impala.admission-control.pool-default-query-options.root.default</name>
    <value>mem_limit=128m,query_timeout_s=20,max_io_buffers=10</value>
  </property>
  <property>
    <name>impala.admission-control.pool-queue-timeout-ms.root.default</name>
    <value>30000</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>impala.admission-control.pool-default-query-options.root.development</name>
    <value>mem_limit=256m,query_timeout_s=30,max_io_buffers=10</value>
  </property>
  <property>
    <name>impala.admission-control.pool-queue-timeout-ms.root.development</name>
    <value>15000</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>
<!--
       Default query options for the 'root.production' pool.
       THIS IS A NEW PARAMETER in CDH 5.7 / Impala 2.5.
       Note that the MEM_LIMIT query option still shows up in here even though it is a
       separate box in the UI. We do that because it is the most important query option

       that people will need (everything else is somewhat advanced).

       MEM_LIMIT takes a per-node memory limit which is specified using one of the
following:
        - '<int>[bB]?'  -> bytes (default if no unit given)
        - '<float>[mM(bB)]' -> megabytes
        - '<float>[gG(bB)]' -> in gigabytes
        E.g. 'MEM_LIMIT=12345' (no unit) means 12345 bytes, and you can append m or g
             to specify megabytes or gigabytes, though that is not required.
-->
  <property>
    <name>impala.admission-control.pool-default-query-options.root.production</name>
    <value>mem_limit=386m,query_timeout_s=30,max_io_buffers=10</value>
  </property>
<!--
  Default queue timeout (ms) for the pool 'root.production'.
  If this isn't set, the process-wide flag is used.
  THIS IS A NEW PARAMETER in CDH 5.7 / Impala 2.5.
-->
  <property>
    <name>impala.admission-control.pool-queue-timeout-ms.root.production</name>
    <value>30000</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. 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 throughput could decrease due to queries spilling to disk or contending for resources; or queries could be cancelled if they exceed the `MEM_LIMIT` setting while running.

In `impala-shell`, you can also specify which resource pool to direct queries to by setting the REQUEST_POOL query option.

If you set up different resource pools for different users and groups, consider reusing any classifications 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).

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 343.

Enabling and Disabling Impala Admission Control Using Cloudera Manager

You perform this task when you have determined that the Impala workload is heavy enough to cause capacity problems on the cluster. The capacity issues could be because of a high volume of concurrent queries, because of heavy-duty join and aggregation queries that require large amounts of memory, or because Impala is being used alongside other Hadoop data management components and the resource usage of Impala must be constrained to work well in a multitenant deployment.

> **Important:**
>
> In CDH 5.8 / Impala 2.6 and higher, admission control and dynamic resource pools are enabled by default.

1. Go to the Impala service.
2. Click the **Configuration** tab.
3. Select **Category** > **Admission Control**.
4. Select or clear both the **Enable Impala Admission Control** checkbox and the **Enable Dynamic Resource Pools** checkbox.
5. Click **Save Changes** to commit the changes.
6. Restart the Impala service.

After completing this task, for further configuration settings, customize the configuration settings for the dynamic resource pools, as described in Creating an Impala Dynamic Resource Pool on page 328 and Editing Dynamic Resource Pools on page 330.

Configuring Impala Admission Control Using Cloudera Manager

In CDH 5.7 / Impala 2.5 and higher, Cloudera recommends configuring admission control at a granular level, using the dynamic resource pools feature, rather than using the cluster-wide settings.

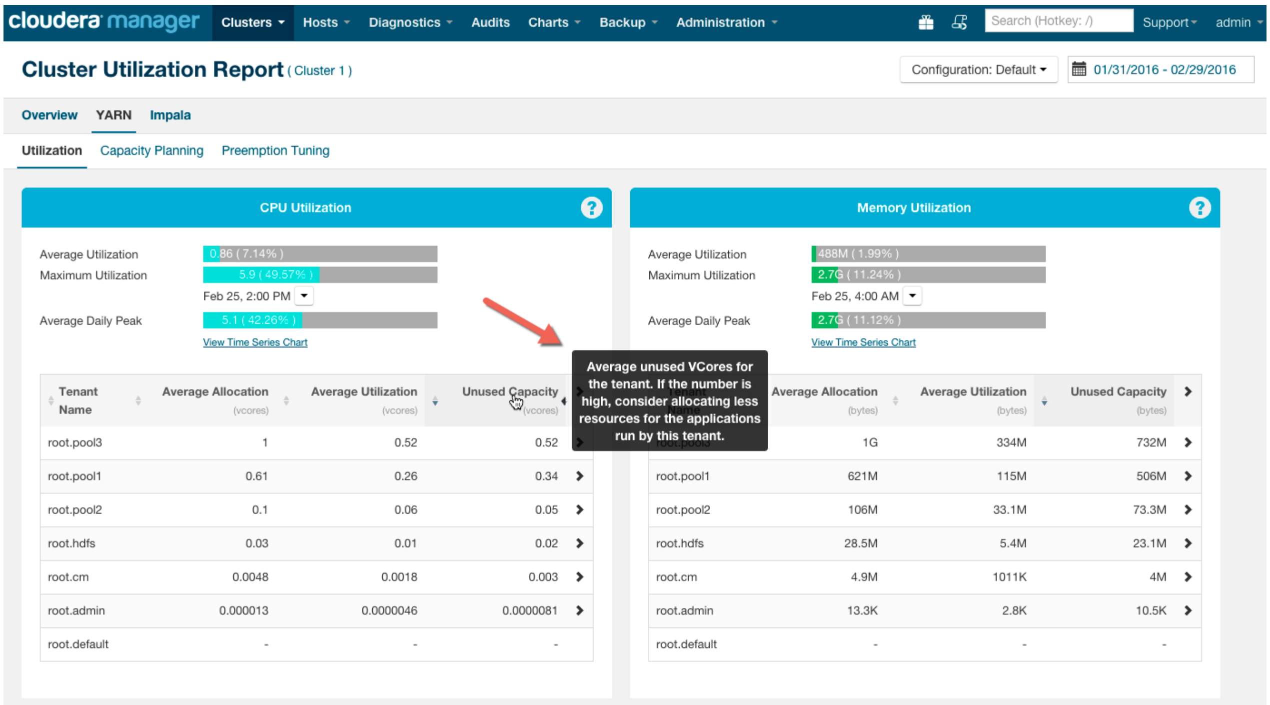
- In the **Admission Control** configuration group for Impala, ignore all the settings below the checkboxes to enable or disable admission control and dynamic resource pools.
- Whether you are creating a new pool or editing the settings of an existing one, follow the decision process in Choosing Settings for an Impala Dynamic Resource Pool on page 328.
- If you are subdividing your workload and do not already have a dynamic resource pool set up for it, follow the procedure in Creating an Impala Dynamic Resource Pool on page 328.
- To edit the properties of an existing dynamic resource pool, follow the procedure in Editing Dynamic Resource Pools on page 330.

Cluster Utilization Reports

The **Cluster Utilization Report** screens in Cloudera Manager display aggregated utilization information for YARN and Impala jobs. The reports display CPU utilization, memory utilization, resource allocations made due to the YARN fair scheduler, and Impala queries. The report displays aggregated utilization for the entire cluster and also breaks out utilization by *tenant*, which is either a user or a resource pool. You can configure the report to display utilization for a range of dates, specific days of the week, and time ranges.

The report displays the current utilization of CPU and memory resources and the resources that were allocated using the Cloudera Manager resource management features. See Resource Management on page 320.

Using the information displayed in the **Cluster Utilization Report**, a CDH cluster administrator can verify that sufficient resources are available for the number and types of jobs running in the cluster. An administrator can use the reports to tune resource allocations so that resources are used efficiently and meet business requirements. Tool tips in the report pages provide suggestions about how to improve performance based on the information displayed in the report. Hover over a label to see these suggestions and other information. For example:



You can tune the following:

- CPU and memory allocations
- Weights for each pool
- Scheduling rules
- Preemption thresholds
- Maximum number of running and queued Impala queries
- Maximum timeout for the queue of Impala queries
- Placement rules
- Number of hosts in a cluster
- Memory capacity of hosts
- Impala Admission Control pool and queue configurations

> **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 583.

The **Cluster Utilization Report** is only available with Cloudera Manager 5.7 and higher and CDH 5.7 and higher.

If you want to create your own reports with similar functionality, or if you want to export the report results, see Creating a Custom Cluster Utilization Report on page 360.

Configuring the Cluster Utilization Report

This topic describes the prerequisites and configurations required to use the **Cluster Utilization Report**.

Enabling the Cluster Utilization Report

By default, the **Cluster Utilization Report** displays aggregated CPU and memory utilization for an entire CDH cluster and for YARN and Impala utilization. You can also view this utilization by *tenants*, which include Linux users and Dynamic Resource Pools on page 326. To see utilization for a tenant, you must configure the tenant and define resource limits for it.

You must configure several parameters to enable the **Cluster Utilization Report**:

1. Enable YARN utilization metrics collection.
    a. Go to the YARN Service
    b. Click the **Configuration** tab.
    c. Select **Category** > **Monitoring**.
    d. Type `container` in the **Search** box.
    e. Select the **Enable Container Usage Metrics Collection** parameter.
    f. Enter a username for the MapReduce job that collects the metrics in the **Container Usage MapReduce Job User** parameter. The username you enter must be a Linux user on all the cluster hosts. If you are using an Active Directory KDC, the username must also exist in Active Directory. For secure clusters, the user must not be banned or below the minimum user ID. You can view the list of banned users (`banned.users`) and the minimum user ID (`min.user.id`) by selecting **Clusters** > **<*YARN service*>** > **Configuration**.

    > **Note:** The user that is configured with the **Container Usage MapReduce Job User** property in the YARN service requires permissions to read the subdirectories of the HDFS directory specified with the **Cloudera Manager Container Usage Metrics Directory** property. The default umask of `022` allows any user to read from that directory. However, if a more strict umask (for example, `027`) is used, then those directories are not readable by any user. In that case the user specified with the **Container Usage MapReduce Job User** property should be added to the same group that owns the subdirectories.
    >
    > For example, if the `/tmp/cmYarnContainerMetrics/20161010` subdirectory is owned by user and group `yarn:hadoop`, the user specified in **Container Usage MapReduce Job User** should be added to the hadoop group.

    > **Note:** The directories you specify with the **Cloudera Manager Container Usage Metrics Directory** and **Container Usage Output Directory** properties should not be located in encryption zones.

    g. (Optional) Enter the resource pool in which the container usage collection MapReduce job runs in the **Container Usage MapReduce Job Pool** parameter. Cloudera recommends that you dedicate a resource pool for running this MapReduce job.

    > **Note:** If you specify a custom resource pool, ensure that the placement rules for the cluster allow for it. The first rule must be for resource pools to be specified at run time with the **Create pool if it does not exist** option selected. Alternatively, ensure that the pool you specify already exists. If the placement rule is not properly configured or the resource pool does not already exist, the job may run in a different pool.

    h. Click **Save Changes** to commit the changes.
    i. Click the **Actions** drop-down list and select **Create CM Container Usage Metrics Dir**.
    j. Restart the YARN service:

a. Go to the YARN service.
b. Select **Actions** > **Restart**.

2. Enable Impala utilization collection.

    a. Go to the Impala service.
    b. Click the **Configuration** tab.
    c. Select **Category** > **Admission Control**.
    d. Select or clear both the **Enable Impala Admission Control** checkbox and the **Enable Dynamic Resource Pools** checkbox.
    e. Click **Save Changes** to commit the changes.
    f. Restart the Impala service.

Configuring the Cluster Utilization Report

To access the **Cluster Utilization Report**, go to **Clusters** and then select **Utilization Report** for the cluster. The **Overview** tab displays when you first open the report.

The upper-right part of the page has two controls that you use to configure the **Cluster Utilization Report**:

You can apply a configuration and date range that applies to all tabs in the report:

1. Click the **Configuration** drop-down menu.
2. Select one the configured options, or create a new configuration:

    a. Click **Create New Configuration**.
    b. Enter a **Configuration Name**.
    c. Select the **Tenant Type**, either **Pool** or **User**.
    d. Select the days of the week for which you want to report utilization.
    e. Select **All Day**, or use the drop-down menus to specify a utilization time range for the report.
    f. Click **Create**.

    The configuration you created is now available from the **Configuration** drop-down menu.

3. Select a date range for the report:

    a. Click the date range button.
    b. Select one of the range options (**Today**, **Yesterday**, **Last 7 Days**, **Last 30 Days**, or **This Month**) or click **Custom Range** and select the beginning and ending dates for the date range.

Using the Cluster Utilization Report to Manage Resources

To access the **Cluster Utilization Report**, go to **Clusters** and then select **Utilization Report** for the cluster. The **Overview** tab of the report displays.

> **Note:** The report updates utilization information every hour. The utilization information for Impala and YARN queries does not display in the **Cluster Utilization Report** until captured by the hourly update.

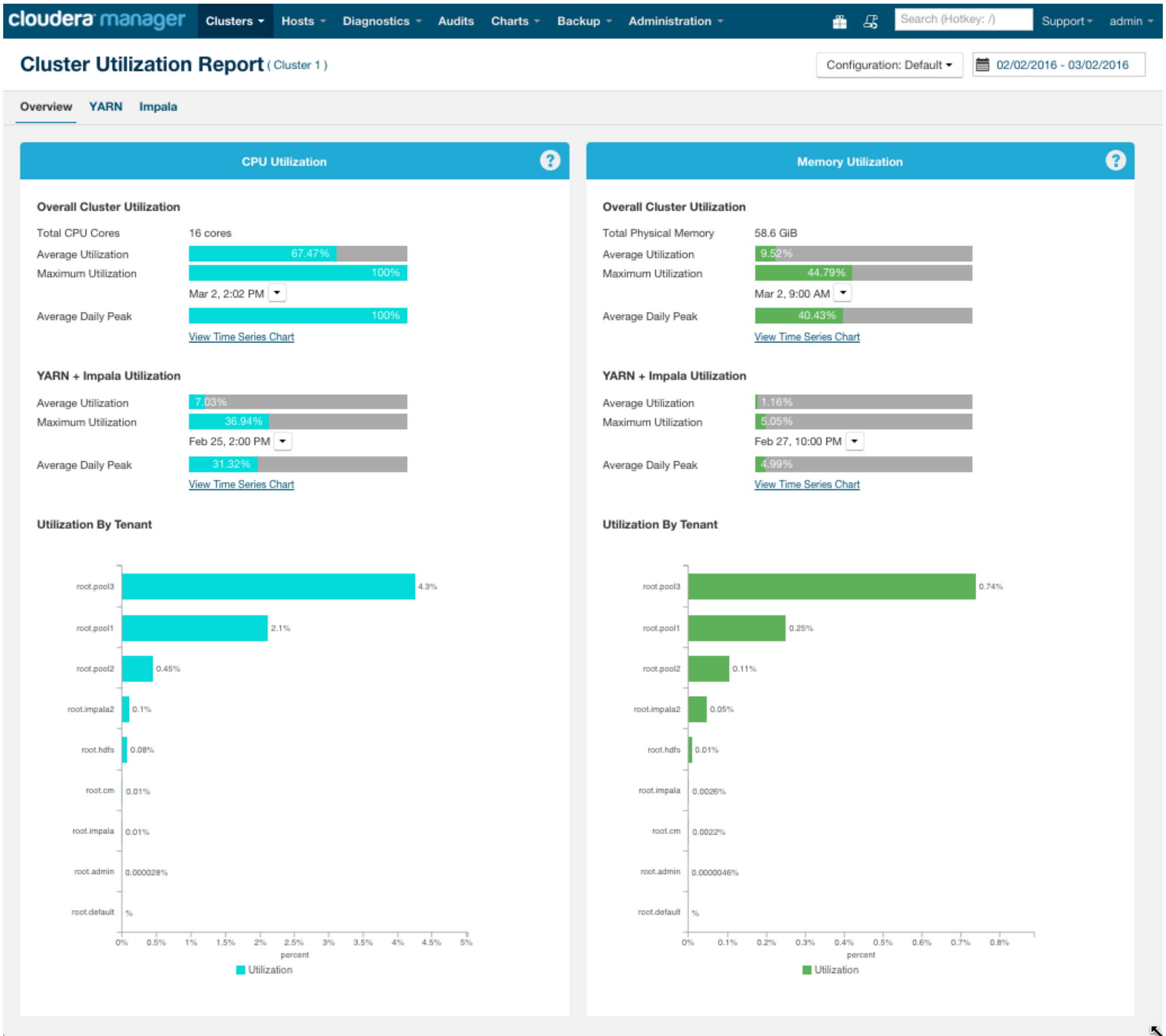


Figure 12: Cluster Utilization Report Overview Tab

The Cluster Utilization Report is divided into the following tabs:

- Overview Tab on page 355
- YARN Tab on page 357
- Impala Tab on page 358

Overview Tab

The **Overview** tab provides a summary of CPU and memory utilization for the entire cluster and also for only YARN applications and Impala queries. Two sections, **CPU Utilization** and **Memory Utilization**, display the following information:

CPU Utilization	Memory Utilization
Overall Cluster Utilization • **Total CPU Cores** – Average number of CPU cores available during the reporting window. • **Average Utilization** – Average CPU utilization for the entire cluster, including resources consumed by user applications and CDH services.	**Overall Cluster Utilization** • **Total Physical Memory** – Average physical memory available in the cluster during the reporting window. • **Average Utilization** – Average memory consumption for the entire cluster, including resources consumed by user applications and CDH services.

<table>
<tr><th>CPU Utilization</th><th>Memory Utilization</th></tr>
<tr>
<td>

Maximum Utilization – Maximum CPU utilization for the entire cluster during the reporting window, including resources consumed by user applications and CDH services. If this value is high, consider adding more hosts to the cluster.
Click the drop-down menu next to the date and select one of the following to view details about jobs running when maximum utilization occurred:

View YARN Applications Running at the Time
View Impala Queries Running at the Time

Average Daily Peak – Average daily peak CPU consumption for the entire cluster during the reporting window. This includes resources consumed by user applications and CDH services. The number is computed by averaging the maximum resource consumption for each day of the reporting period.
Click View Time Series Chart to view a chart of peak utilization.

</td>
<td>

Maximum Utilization – Maximum memory consumption for the entire cluster during the reporting window, including resources consumed by user applications and CDH services. If this value is high, consider adding more hosts to the cluster.
Click the drop-down menu next to the date and select one of the following to view details about jobs running when maximum utilization occurred:

View YARN Applications Running at the Time
View Impala Queries Running at the Time

Average Daily Peak – Average daily peak memory consumption for the entire cluster during the reporting window, including resources consumed by user applications and CDH services. The number is computed by averaging the maximum memory utilization for each day of the reporting period.
Click View Time Series Chart to view a chart of peak utilization.

</td>
</tr>
<tr>
<td>
YARN + Impala Utilization

Average Utilization – Average resource consumption by YARN applications and Impala queries that ran on the cluster.
Maximum Utilization – Maximum resource consumption by YARN applications and Impala queries that ran on the cluster.
Click the drop-down menu next to the date and select one of the following to view details about jobs running when maximum utilization occurred:

View YARN Applications Running at the Time
View Impala Queries Running at the Time

Average Daily Peak – Average daily peak resource consumption by YARN applications and Impala queries during the reporting window. The number is computed by finding the maximum resource consumption per day and calculating the mean.
Click View Time Series Chart to view a chart of peak utilization.

</td>
<td>
YARN + Impala Utilization

Average Utilization – Average memory consumption by YARN applications and Impala queries that ran on the cluster.
Maximum Utilization – Maximum memory consumption for the entire cluster during the reporting window, including resources consumed by user applications and CDH services. If this is high, consider adding more hosts to the cluster.
Click the drop-down menu next to the date and select one of the following to view details about jobs running when maximum utilization occurred:

View YARN Applications Running at the Time
View Impala Queries Running at the Time

Average Daily Peak – Average daily peak memory consumption by YARN applications and Impala queries during the reporting window. The number is computed by finding the maximum resource consumption per day and then calculating the mean.
Click View Time Series Chart to view a chart of peak utilization.

</td>
</tr>
<tr>
<td>
Utilization by Tenant
Displays overall utilization for each tenant. Tenants can be either pools or users. See Configuring the Cluster Utilization Report on page 354.
</td>
<td>
Utilization by Tenant
Displays overall utilization for each tenant. Tenants can be either pools or users. See Configuring the Cluster Utilization Report on page 354.
</td>
</tr>
</table>

YARN Tab

The **YARN** tab displays CPU and memory utilization for YARN applications on three tabs:

- Utilization Tab on page 357
- Capacity Planning Tab on page 358
- Preemption Tuning Tab on page 358

For information about managing YARN resources, see:

- YARN (MRv2) and MapReduce (MRv1) Schedulers on page 337
- Enabling and Disabling Fair Scheduler Preemption on page 341
- Dynamic Resource Pools on page 326

Utilization Tab

Table 24: Utilization Tab

<table>
<tr><th>CPU Utilization</th><th>Memory Utilization</th></tr>
<tr><td>Summary section:
Average Utilization – Average number of vcores used by YARN applications. The percentage reported is of the total number of vcores configured for YARN.
Maximum Utilization – Maximum number of vcores used by YARN applications. The percentage reported is of the total number of vcores configured for YARN.

Click the drop-down menu next to the date and select View YARN Applications Running at the Time to view details about jobs running when maximum utilization occurred.
Average Daily Peak – Average daily peak vcores used by YARN applications. The number is computed by finding the maximum resource consumption per day and calculating the mean. The percentage reported is of the total number of vcores configured for YARN.

Click View Time Series Chart to view a chart of peak utilization.</td><td>Summary section:
Average Utilization – Average memory used by YARN applications. The percentage reported is of the total container memory configured for YARN.
Maximum Utilization – Maximum memory used by YARN applications. The percentage reported is of the total container memory configured for YARN.

Click the drop-down menu next to the date and select View YARN Applications Running at the Time to view details about jobs running when maximum utilization occurred.
Average Daily Peak – Average daily peak memory used by YARN applications. The number is computed by finding the maximum resource consumption per day and calculating the mean. The percentage reported is of the total container memory configured for YARN.

Click View Time Series Chart to view a chart of peak utilization.</td></tr>
<tr><td>Utilization by Tenant

Displays overall utilization for each tenant. The tenants can be either pools or users. See Configuring the Cluster Utilization Report on page 354

Utilization by tenant is displayed in a table with the following columns:
Tenant Name
Average Allocation – Average number of vcores allocated to YARN applications of the tenant. The percentage reported is of the total number of vcores configured for YARN.
Average Utilization – Average number of vcores used by YARN applications. The percentage reported is of the total number of vcores configured for YARN.</td><td>Utilization by Tenant

Displays overall utilization for each tenant. The tenants can be either pools or users. See Configuring the Cluster Utilization Report on page 354.

Utilization by tenant is displayed in a table with the following columns:
Tenant Name
Average Allocation – Average memory allocated to YARN applications of the tenant. The percentage reported is of the total container memory configured for YARN.
Average Utilization – Average memory used by YARN applications. The percentage reported is of the total container memory configured for YARN.</td></tr>
</table>

CPU Utilization	Memory Utilization
• **Unused Capacity** – Average unused vcores for the tenant. If this number is high, consider allocating less resources for the applications run by this tenant. Click a column header to sort the table by that column. Click the › icon in the header row of the table to view utilization charts for all tenants. Click › in a row to view CPU utilization for a single tenant.	• **Unused Capacity** – Average unused memory for the tenant. If this number is high, consider allocating less resources for the applications run by this tenant. Click a column header to sort the table by that column. Click the › icon in the header row of the table to view utilization charts for all tenants. Click › in a row to view CPU utilization for a single tenant.

Adjusting YARN Resources

To adjust YARN resources. Go to the YARN service and select **Configuration** > **Category** > **Resource Management** and configure the following properties:

- vcores: **Container Virtual CPU Cores**
- Memory: **Container Memory**

Capacity Planning Tab

The **Capacity Planning Tab** displays a table showing how the weights assigned to YARN Dynamic Resource Pools affect CPU and memory allocations. The table displays the following columns:

- **Tenant Name**
- **CPU Steady Fair Share** – Displays the average number of CPU vcores allocated for each tenant based on the weights assigned to dynamic resource pools.
- **Memory Steady Fair Share** – Displays the average memory allocated for each tenant based on the weights assigned to dynamic resource pools.
- **Wait Ratio During Contention** – The wait ratio is the average percentage of containers in the YARN pool that were pending when there was at least one pending container in the pool. If a pool running critical applications has a high wait ratio, consider increasing the weight of that pool. If several pools in the cluster have a high wait ratio, consider adding more hosts to the cluster.

Click a column header to sort the table by that column.

Preemption Tuning Tab

The **Preemption Tuning** tab displays graphs for each tenant that display the average steady fair share allocations against the average instantaneous fair share allocations and average overall allocations for CPU and memory allocations.

The **CPU** section shows the average allocated vcores, instantaneous fair share of vcores, and steady fair share of vcores whenever the YARN pool was facing contention with resources (times when there was at least one pending container). If the allocated vcores are less than the steady fair share during contention, consider making preemption more aggressive by doing the following:

- Enable fair scheduler preemption.
- Reduce the fair scheduler preemption utilization threshold.
- If you have configured a preemption timeout for a pool (on the **Dynamic Resource Pool Configuration** page (**Clusters** > ***cluster name*** > **Resource Management** > **Dynamic Resource Pool**), reduce the length of the timeout for pools with a high wait ratio. See Dynamic Resource Pools on page 326.

See Enabling and Disabling Fair Scheduler Preemption on page 341.

The **Memory** section shows the average allocated memory, instantaneous fair share of memory, and steady fair share of memory whenever the YARN pool was facing contention with resources (times when there was at least one pending container). If the allocated memory is less than the Steady Fair Share during contention, consider making preemption more aggressive, as described previously for CPU.

Impala Tab

The **Impala** tab displays CPU and memory utilization for Impala queries using three tabs:

- Queries Tab on page 359
- Peak Memory Usage Tab on page 359
- Spilled Memory Tab on page 360

For information about managing Impala resources, see:

- Admission Control and Query Queuing on page 343
- Managing Impala Admission Control on page 351

Queries Tab

The **Overview** tab displays information about Impala queries.

The top part of the page displays summary information about Impala queries for the entire cluster. The table in the lower part displays the same information by tenant. Both sections display the following:

- **Total** – Total number of queries.

  Click the link with the total to view details and charts about the queries.

- **Avg Wait Time in Queue** – Average time, in milliseconds, spent by a query in an Impala pool while waiting for resources. If this number is high, consider increasing the resources allocated to the pool. If this number is high for several pools, consider increasing the number of hosts in the cluster.
- **Successful** – The number and percentage of queries that finished successfully.

  Click the link with the total to view details and charts about the queries.

- **Memory Limit Exceeded** – Number and percentage of queries that failed due to insufficient memory. If there are such queries, consider increasing the memory allocated to the pool. If there are several pools with such queries, consider increasing the number of hosts in the cluster.
- **Timed Out in Queue** – Number of queries that timed out while waiting for resources in a pool. If there are such queries, consider increasing the maximum number of running queries allowed for the pool. If there are several pools with such queries, consider increasing the number of hosts in the cluster.
- **Rejected** – Number of queries that were rejected by Impala because the pool was full. If this number is high, consider increasing the maximum number of queued queries allowed for the pool. See Admission Control and Query Queuing on page 343.

Click a column header to sort the table by that column.

Peak Memory Usage Tab

This report shows how Impala consumes memory at peak utilization. If utilization is high for a pool, consider adding resources to the pool. If utilization is high for several pools, consider adding more hosts to the cluster.

The **Summary** section of this page displays aggregated peak memory usage information for the entire cluster and the **Utilization by Tenant** section displays peak memory usage by tenant. Both sections display the following:

- **Max Allocated**
  - **Peak Allocation Time** – The time when Impala reserved the maximum amount of memory for queries.

    Click the drop-down list next to the date and time and select **View Impala Queries Running at the Time** to see details about the queries.

  - **Max Allocated** – The maximum memory that was reserved by Impala for executing queries. If the percentage is high, consider increasing the number of hosts in the cluster.
  - **Utilized at the Time** – The amount of memory used by Impala for running queries at the time when maximum memory was reserved.

    Click **View Time Series Chart** to view a chart of peak memory allocations.

  - **Histogram of Allocated Memory at Peak Allocation Time** – Distribution of memory reserved per Impala daemon for executing queries at the time Impala reserved the maximum memory. If some Impala daemons have reserved memory close to the configured limit, consider adding more physical memory to the hosts.

> **Note:** This histogram is generated from the minute-level metrics for Impala daemons. If the minute-level metrics for the timestamp at which peak allocation happened are no longer present in the Cloudera Service Monitor Time-Series Storage, the histogram shows no data. To maintain a longer history for the minute-level metrics, increase the value of the **Time-Series Storage** property for the Cloudera Service Monitor. (Go to the **Cloudera Management Service** > **Configuration** and search for **Time-Series Storage**.)

- **Max Utilized**
  - **Peak Usage Time** – The time when Impala used the maximum amount of memory for queries.

    Click the drop-down list next to the date and time and select **View Impala Queries Running at the Time** to see details about the queries.
  - **Max Utilized** – The maximum memory that was used by Impala for executing queries. If the percentage is high, consider increasing the number of hosts in the cluster.
  - **Reserved at the Time** – The amount of memory reserved by Impala at the time when it was using the maximum memory for executing queries.

    Click **View Time Series Chart** to view a chart of peak memory utilization.
  - **Histogram of Utilized Memory at Peak Usage Time** – Distribution of memory used per Impala daemon for executing queries at the time Impala used the maximum memory. If some Impala daemons are using memory close to the configured limit, consider adding more physical memory to the hosts.

> **Note:** This histogram is generated from the minute-level metrics for Impala daemons. If the minute-level metrics for the timestamp at which peak allocation happened are no longer present in the Cloudera Service Monitor Time-Series Storage, the histogram shows no data. To maintain a longer history for the minute-level metrics, increase the value of the **Time-Series Storage** property for the Cloudera Service Monitor. (Go to the **Cloudera Management Service** > **Configuration** and search for **Time-Series Storage**.)

Spilled Memory Tab

The **Spilled Memory** tab displays information about Impala spilled memory. These disk spills can deteriorate the performance of Impala queries significantly. This report shows the amount of disk spills for Impala queries by tenant. If disk spill is high for a pool, consider adding resources to the pool. If disk spill is high for several pools, consider adding more hosts to the cluster.

For each tenant, the following are displayed:

- **Average Spill** – Average spill per query
- **Maximum Spill** – Maximum memory spilled per hour

Creating a Custom Cluster Utilization Report

Cloudera Manager provides a Cluster Utilization Report that displays aggregated utilization information for YARN and Impala jobs. If you wish to export the data from this report, you can build custom reports based on the same metrics data using the Cloudera Manager Admin console or the Cloudera Manager API. This topic describes the metrics and queries you can use to build these custom reports. These reports all use the tsquery Language to chart time-series data.

Metrics and Queries

For more information about the *Data Granularity* described in these metrics, see Metric Aggregation.

Many of the metrics described below use a data granularity of *hourly*. This is not required, but is recommended because some of the YARN utilization metrics are only available hourly and using the hourly granularity allows for consistent reporting.

Cluster-Level CPU and Memory Metrics

Total cluster CPU usage

Data Granularity: hourly

Units: percentage

tsquery:

```
SELECT
   cpu_percent_across_hosts
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Total CPU Cores in the cluster

Data Granularity: hourly

Units: CPU cores

tsquery:

```
SELECT
   total_cores_across_hosts
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Total cluster memory usage

Data Granularity: hourly

Units: percentage

tsquery:

```
SELECT
   100 * total_physical_memory_used_across_hosts/total_physical_memory_total_across_hosts
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Total cluster memory usage

Time series of total cluster memory usage.

Data Granularity:hourly

Units: Byte seconds

tsquery:

```
SELECT
   total_physical_memory_total_across_hosts
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

CPU used by Impala

Time series of total Impala CPU usage in milliseconds.

Data Granularity: hourly

Units: milliseconds

tsquery:

```
SELECT
   counter_delta(impala_query_thread_cpu_time_rate)
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Memory used by Impala

Time series of Impala memory usage

Data Granularity: hourly

Units: byte seconds

tsquery:

```
SELECT
   counter_delta(impala_query_memory_accrual_rate)
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

CPU used by YARN

The `yarn_reports_containers_used_cpu` metric used in this tsquery is generated per hour, therefore the data granularity used for this query is the raw metric value.

Data Granularity: Raw

Units: percent seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_cpu FROM REPORTS
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

Memory used by YARN

Yarn memory usage. The `yarn_reports_containers_used_memory` metric used in this tsquery is generated per hour, therefore the data granularity used for this query is the raw metric value.

Data Granularity: raw metric value

Units: megabyte seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_memory
FROM
   REPORTS
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

Pool-Level CPU and Memory Metrics

CPU used by Impala pool

CPU usage for an Impala pool.

Data Granularity: hourly

Units: milliseconds

tsquery:

```
SELECT
   counter_delta(impala_query_thread_cpu_time_rate)
WHERE
   category=IMPALA_POOL
   AND poolName=Pool_Name
```

Memory used by Impala pool

Data Granularity: hourly

Units: byte seconds

tsquery:

```
SELECT
   counter_delta(impala_query_memory_accrual_rate)
WHERE
   category=IMPALA_POOL
   AND poolName=Pool_Name
```

CPU used by YARN pool

Provides CPU metrics per YARN pool and user. You can aggregate a pool-level metric from this query.

Data Granularity: Raw

Units: percent seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_cpu FROM REPORTS
WHERE
   category=YARN_POOL_USER
```

Memory used by YARN pool

Provides memory metrics per YARN pool and user. You can aggregate a pool-level metric from this query.

Data Granularity: hourly

Units: megabyte seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_memory
FROM
   REPORTS
WHERE
   category=YARN_POOL_USER
```

YARN Metrics

YARN VCore usage

Data Granularity: Raw

Units: VCore seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_vcores
FROM
```

```
   REPORTS
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

Total VCores available to YARN

Data Granularity: hourly

Units: Number of VCores (Note that this value is not multiplied by the time unit.)

tsquery:

```
SELECT
   total_allocated_vcores_across_yarn_pools + total_available_vcores_across_yarn_pools
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

YARN Memory usage

Data Granularity: Raw

Units: MB seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_memory FROM REPORTS
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

Total memory available to YARN

Data Granularity: hourly

Units: MB (Note that this value is not multiplied by the time unit.)

tsquery:

```
SELECT
   total_available_memory_mb_across_yarn_pools +
total_allocated_memory_mb_across_yarn_pools
WHERE
   category=SERVICE
   AND clusterName=Cluster_Name
```

Pool-level VCore usage

The results of this query return the usage for each user in each pool. To see the total usage for a pool, sum all users of the pool.

Data Granularity: Raw

Units: VCore seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_vcores FROM REPORTS
WHERE
   category=YARN_POOL_USER
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level memory usage

The results of this query return the usage for each user in each pool. To see the total usage for a pool, sum all users of the pool.

Data Granularity: Raw

Units: MB seconds

tsquery:

```
SELECT
   yarn_reports_containers_used_memory FROM REPORTS
WHERE
   category=YARN_POOL_USER
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level allocated VCores

The results of this query return the usage for each user in each pool. To see the total usage for a pool, sum all users of the pool.

Data Granularity: raw metric value

Units: VCore seconds

tsquery:

```
SELECT
   yarn_reports_containers_allocated_vcores FROM REPORTS
WHERE
   category=YARN_POOL_USER
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level allocated memory

The results of this query return the usage for each user in each pool. To see the total usage for a pool, sum all users of the pool.

Data Granularity: raw metric value

Units: megabyte seconds

tsquery:

```
SELECT
   yarn_reports_containers_allocated_memory
FROM
   REPORTS
WHERE
   category=YARN_POOL_USER
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level steady fair share VCore

Data Granularity: hourly

Units: VCores

tsquery:

```
SELECT
   steady_fair_share_vcores
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level fair share VCore

Data Granularity: hourly

Units: VCores

tsquery:

```
SELECT
   fair_share_vcores
WHERE
   category=YARN_POOL
```

Pool-level steady fair share memory

Data Granularity: hourly

Units: MB

tsquery:

```
SELECT
   steady_fair_share_mb
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool-level fair share memory

Data Granularity: hourly

Units: MB

tsquery:

```
SELECT
   fair_share_mb
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Metric indicating contention

Data Granularity: hourly

Units: percentage

tsquery:

```
SELECT
   container_wait_ratio
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

YARN Contention-Related Metrics

Use the following metrics to monitor resource contention.

Pool-level allocated VCores when contention occurs

Data Granularity: hourly

Units: VCores

tsquery:

```
SELECT
   allocated_vcores_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool level steady fair share VCores when contention occurs

Data Granularity: hourly

Units: VCores

tsquery:

```
SELECT
   steady_fair_share_vcores_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool level fair share VCores when contention occurs

Data Granularity: hourly

Units: VCores

tsquery:

```
SELECT
   fair_share_vcores_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool level allocated memory when contention occurs

Data Granularity: hourly

Units: MB

tsquery:

```
SELECT
   allocated_memory_mb_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool level steady fair share memory when contention occurs

Data Granularity: hourly

Units: MB

tsquery:

```
SELECT
   steady_fair_share_mb_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Pool level fair share memory when contention occurs

Data Granularity: hourly

Units: MB

tsquery:

```
SELECT
   fair_share_mb_with_pending_containers
WHERE
   category=YARN_POOL
```

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Impala-Specific Metrics

To view metrics for a specific pool, add `poolName=`*`Pool Name`* to the tsquery statement.

Total reserved memory

Data Granularity: hourly

Units: MB seconds

tsquery:

```
SELECT
  total_impala_admission_controller_local_backend_mem_reserved_across_impala_daemon_pools
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Total used memory

Data Granularity: hourly

Units: MB seconds

tsquery:

```
SELECT
  total_impala_admission_controller_local_backend_mem_usage_across_impala_daemon_pools

 WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

Total available memory

Data Granularity: hourly

Units: MB seconds

tsquery:

```
SELECT
   total_mem_tracker_process_limit_across_impalads
WHERE
   category=CLUSTER
   AND clusterName=Cluster_Name
```

> **Note:** To query for pool-level metrics, change the `category` to `IMPALA-POOL` in the above tsquery statements.

Impala Query Counter Metrics

Include the following in the SELECT statement of the tsquery to get information about the rate of Impala queries:

- `counter_delta(queries_ingested_rate)`
- `counter_delta(queries_successful_rate)`
- `counter_delta(queries_rejected_rate)`
- `counter_delta(queries_oom_rate)`
- `counter_delta(queries_timed_out_rate)`
- `counter_delta(impala_query_admission_wait_rate)`
- `counter_delta(impala_query_memory_spilled_rate)`

For example:

```
SELECT
   counter_delta(queries_ingested_rate)
 WHERE
   category=IMPALA_POOL
   AND clusterName=Cluster_Name
   AND serviceName=Service_Name
```

Calculations for reports

All the metrics listed in this topic return a time series of metric values. Depending on the collection frequency of the metric itself and the data granularity you use when issuing tsquery statements, the results return metric values in different frequencies and therefore there are different ways to handle the metric values.

Note the following about how to correctly perform calculations using metric values:

- YARN container metrics are generated once per hour resulting in one raw metric value every hour. Therefore, the most detailed results possible for YARN CPU and memory usage are hourly reports.
- Hourly aggregates are summarized from raw metric values. These aggregates include a set of statistics that include the sum, maximum, minimum, count and other statistics that summarize the raw metric values. When you use the hourly granularity, you lose the single values of the raw metric values. However, you can still get peak usage data for such metrics. For more information, see Metric Aggregation.
- For some of the YARN metrics described in this topic, the tsquery statement aggregates from the pool and user level to pool level in the Cloudera Manager Cluster Utilization reports. For these queries, because the maximum and minimum for different pool and user combinations are not likely to happen at the same time, there is no way to get the peak usage across pool and user combinations, or at the pool level. The only meaningful results possible are average and sum.
- When calculating CPU/Memory usage percentage, pay attention to the units for each metric. For example, if the cluster consistently has 8 VCores, the total VCore seconds for each hour would be 8 * 3600 VCore seconds. You can then use this adjusted number to compare with the VCore seconds used by YARN or YARN pools.

Retrieving metric data using the Cloudera Manager API

There is a Time series endpoint exposed by the Cloudera Manager REST API. See Cloudera Manager API documentation timeseries Endpoint. The API accepts tsquery statements as input for which metrics need to be retrieved during the specified time window. The API provides functionality to specify the desired data granularity (for example, raw metric values, TEN_MINUTES, HOURLY etc.). Each granularity level of data is maintained in a leveldb table (see Data Granularity and Time-Series Metric Data). This data is aggregated from raw metric values such as minimum, maximum, etc. within the corresponding data window.

For example, if you do not need the metric data at a specific timestamp but care more about the hourly usage, HOURLY data should be good enough. In general, the longer the granular window it is, the less storage it is taking, and thus the longer period of time you are able to keep that level of data without being purged when the storage hits the configured limit. In the case of Cloudera Manager Cluster Utilization Reports, Cloudera Manager generates the reports based on an hourly window.

To view the Cloudera Manager Service Monitor data storage granularities, go to **Clusters** > **Cloudera Management Service** > **Service Monitor** > **Charts Library** > **Service Monitor Storage** and scroll down to see the **Data Duration Covered** table to see the earliest available data points for each level of granularity. The value in the **last(duration_covered)** column indicates the age of the oldest data in the table.

Data Duration Covered

September 14, 2016 3:40 PM

Search

Entity	last(duration_covered)
impala-query-monitoring - profiles (RAW)	9.4h
impala-query-monitoring - profiles_end_time (RAW)	9.4h
impala-query-monitoring - queries (RAW)	9.4h
service-monitoring - reports_stream (RAW)	9.4h
service-monitoring - reports_type (RAW)	9.4h
service-monitoring - stream (RAW)	9.4h
service-monitoring - subject_ts (RAW)	9.4h
service-monitoring - ts_stream_rollup_PT21600S (SIX_HOURLY)	9.4h
service-monitoring - ts_stream_rollup_PT3600S (HOURLY)	9.4h
service-monitoring - ts_stream_rollup_PT600S (TEN_MINUTELY)	9.4h
service-monitoring - ts_stream_rollup_PT604800S (WEEKLY)	9.4h
service-monitoring - ts_stream_rollup_PT86400S (DAILY)	9.4h
service-monitoring - ts_subject (RAW)	9.4h
service-monitoring - ts_type_rollup_PT21600S (SIX_HOURLY)	9.4h
service-monitoring - ts_type_rollup_PT3600S (HOURLY)	9.4h
service-monitoring - ts_type_rollup_PT600S (TEN_MINUTELY)	9.4h
service-monitoring - ts_type_rollup_PT604800S (WEEKLY)	9.4h
service-monitoring - ts_type_rollup_PT86400S (DAILY)	9.4h
service-monitoring - type (RAW)	9.4h
yarn-application-monitoring - application_details (RAW)	9.4h
yarn-application-monitoring - applications (RAW)	9.4h
yarn-application-monitoring - applications_end_time (RAW)	9.4h

To configure the Time series storage used by the Service Monitor, go to **Clusters** > **Cloudera Management Service** > **Configuration** > **Charts Library** > **Service Monitor Storage** and search for "Time-Series Storage".

Querying metric data using the Cloudera Manager Admin Console

To build charts that query time series data using the Cloudera Manager Admin console, go to **Charts** > **Chart Builder**. When building charts, it may be useful to choose the data granularity by clicking the **Show additional options** link on the chart builder page and then selecting the **Data Granularity** drop-down list. See Charting Time-Series Data.

Selecting data granularity in chart builder:

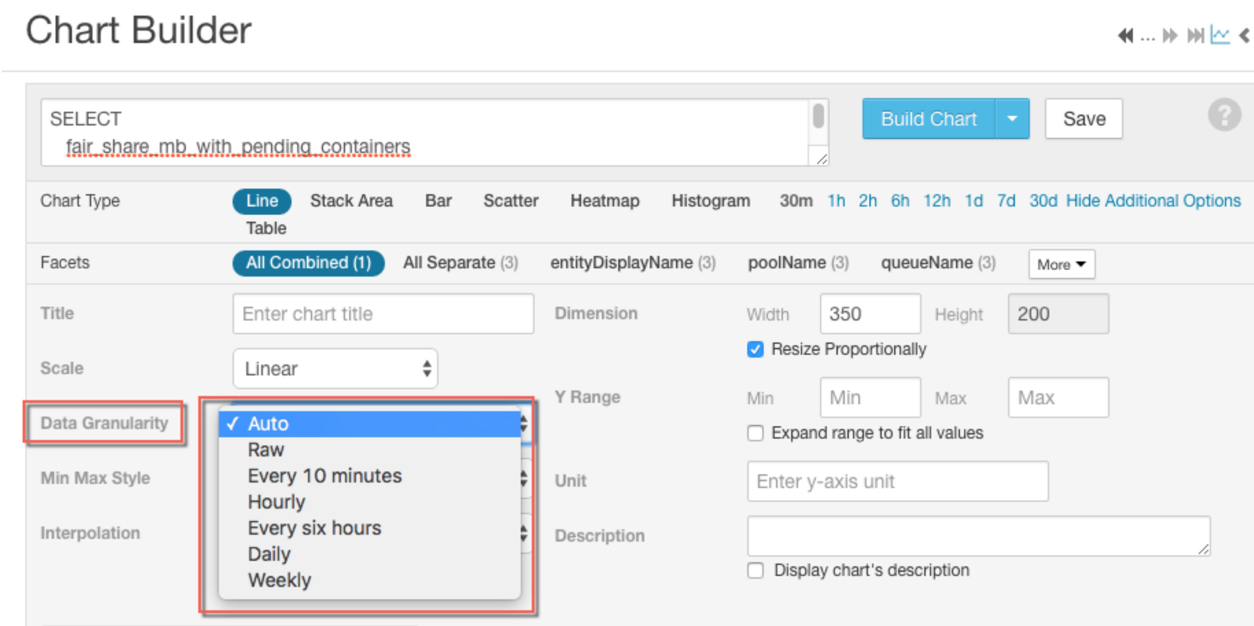


Chart Builder
SELECT
fair_share_mb_with_pending_containers
Build Chart
Save
Chart Type
Line
Stack Area
Bar
Scatter
Heatmap
Histogram
Table
30m 1h 2h 6h 12h 1d 7d 30d Hide Additional Options
Facets
All Combined (1)
All Separate (3)
entityDisplayName (3)
poolName (3)
queueName (3)
More
Title
Enter chart title
Dimension
Width
350
Height
200
Resize Proportionally
Scale
Linear
Y Range
Min
Max
Expand range to fit all values
Data Granularity
✓ Auto
Raw
Every 10 minutes
Hourly
Every six hours
Daily
Weekly
Min Max Style
Interpolation
Unit
Enter y-axis unit
Description
Display chart's description

High Availability

This guide is for Apache Hadoop system administrators who want to enable continuous availability by configuring clusters without single points of failure.

Not all Hadoop components currently support highly availability configurations. However, some currently SPOF (single point of failure) components can be configured to restart automatically in the event of a failure (**Auto-Restart Configurable**, in the table below). Some components support high availability implicitly because they comprise distributed processes (identified with an asterisk (*) in the table). In addition, some components depend on external databases which must also be configured to support high availability.

High Availability	Auto-Restart Configurable	Components with External Databases
Alert Publisher	Hive Metastore (not possible with Sentry enabled)	Activity Monitor
Cloudera Manager Agent*	Impala catalog service	Cloudera Navigator Audit Server
Cloudera Manager Server	Impala statestore	Cloudera Navigator Metadata Server
Data Node*	Sentry Service	Hive Metastore Server
Event Server	Spark Job History Server	Oozie Server
Flume*	YARN Job History Server	Reports Manager
HBase Master		Sentry Server
Host Monitor		Sqoop Server
Hue (add multiple services, use load balancer)		
Impalad* (add multiple services, use load balancer)		
NameNode		
Navigator Key Trustee		
Node Manager*		
Oozie Server		
RegionServer*		
Reports Manager		
Resource Manager		
Service Monitor		
Solr Server*		
Zookeeper server*		

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 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 host 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 automatic failover to a new NameNode in the case that a host 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).

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 the 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 ensures 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.

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. To achieve this, 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. To ensure this property and prevent the so-called "split-brain scenario," JournalNodes only ever allow a single NameNode to be a writer at a time. During a failover, the NameNode which is to become active simply takes over the role of writing to the JournalNodes, which effectively prevents 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 375.

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

To deploy an HA cluster using Quorum-based Storage, you should prepare the following:

- NameNode hosts - These are the hosts 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 hosts - These are the hosts 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 host, 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 host. You can also run more than three JournalNodes, but 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. Perform all the configuration and setup tasks described under Configuring Hardware for HDFS HA on page 374.
2. Ensure that you have a ZooKeeper service.
3. Go to the HDFS service.
4. 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.
   - The directory owner should be hdfs:hadoop and must have read, write, and execute permission (`drwx------`).

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 stop the dependent services, delete, create, and configure roles and directories as required, create a nameservice and failover controller, restart the dependent services, and deploy the new client configuration.

   > **Important:** Some steps, such as formatting the NameNode may report failure if the action was already completed. However, the configuration steps continue to execute after reporting non-critical failed steps.

5. 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 388.

> **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 Using Cloudera Manager on page 13.
> 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.

Fencing Methods

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 380.

Enabling HDFS HA Using the Command Line

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

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

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.

```
<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 does not 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. 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
`$target_host`	Hostname of the node to be fenced
`$target_port`	IPC port of the node to be fenced
`$target_address`	The above two variables, combined as *host:port*
`$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. For an overview of how automatic failover is implemented, see Automatic Failover on page 373.

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.

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-zkfcs:mypassword` where `hdfs-zkfcs` 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-zkfcs:mypassword
output: hdfs-zkfcs:mypassword->hdfs-zkfcs: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-zkfcs:vlUvLnd8MlacsE80rDuu6ONESbM=: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-zkfcs:vlUvLnd8MlacsE80rDuu6ONESbM=
: 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 374 and [Configuring Software for HDFS HA](#) on page 378, 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 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:**
>
> - Follow these command-line instructions on systems that do not use Cloudera Manager.
> - This information applies specifically to CDH 5.9.x. See Cloudera Documentation for information specific to other releases.

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 377.

     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 388.
2. Uncomment the properties you commented out in Step 2: Unconfigure HA on page 388.
3. Deploy HDFS HA, following the instructions under Deploying HDFS High Availability on page 385.

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 Cloudera Manager

If you configure HBase to use an HA-enabled HDFS instance, Cloudera Manager automatically handles HA configuration for you.

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 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

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

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

$ hive --service metatool -updateLocation hdfs://<new_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 Using Cloudera Manager

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.
6. Restart the Hue service.

Configuring Impala to Work with HDFS HA

1. 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.
2. 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

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.

1. Go to the HDFS service.
2. Click the **Instances** tab.
3. Click **Federation and High Availability**.
4. Locate the row for the Nameservice where you want to fail over the NameNode.
5. Select **Actions** > **Manual Failover**. (This option does not appear if HA is not enabled for the cluster.)
6. 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.

7. 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.

Manually Failing Over to the Standby NameNode 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.

Moving an HA NameNode to a New Host

Moving an HA NameNode to a New Host Using Cloudera Manager

See Moving Highly Available NameNode, Failover Controller, and JournalNode Roles Using the Migrate Roles Wizard on page 169.

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 385.

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` 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 375 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 386, 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
```

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.

Using the dfsadmin Command When HA is Enabled

By default, applicable `dfsadmin` command options are run against both active and standby NameNodes. To limit an option to a specific NameNode, use the `-fs` option. For example,

To turn safe mode on for both NameNodes, run:

`hdfs dfsadmin -safemode enter`

To turn safe mode on for a single NameNode, run:

```
hdfs dfsadmin -fs hdfs://<host>:<port> -safemode enter
```

For a full list of `dfsadmin` command options, run: `hdfs dfsadmin -help`.

Converting From an NFS-mounted Shared Edits Directory to Quorum-based Storage

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.

Converting From an NFS-mounted Shared Edits Directory to 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 high availability, see Enabling HDFS HA Using Cloudera Manager on page 375.

Before you start, make note of the name of the active NameNode role instance. You can find the list of NameNode instances on the **Instances** tab for the HDFS service in the Cloudera Manager Admin Console.

Complete the following steps to change the NameService name for HDFS with HA:

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 on an active **NameNode** role instance.

13. Select **Actions** > **Initialize Shared Edits Directory**.
14. Click the Cloudera Manager logo to return to the **Home** page.
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 is responsible for tracking the resources in a cluster and scheduling applications (for example, MapReduce jobs). Before CDH 5, the ResourceManager was a single point of failure in a YARN cluster. The ResourceManager high availability (HA) feature adds redundancy in the form of an active-standby ResourceManager pair to remove this single point of failure. Furthermore, upon failover from the active ResourceManager to the standby, the applications can resume from the last state saved to the state store; for example, map tasks in a MapReduce job are not run again if a failover to a new active ResourceManager occurs after the completion of the map phase. 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

ResourceManager HA requires ZooKeeper and HDFS services to be running.

Architecture

ResourceManager HA is implemented by means of an active-standby pair of ResourceManagers. On start-up, each ResourceManager is in the standby state; the process is started, but the state is not loaded. When one of the ResourceManagers is transitioning to the active state, the ResourceManager 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 ResourceManager HA.

ResourceManager Restart

Restarting the ResourceManager allows for the recovery of in-flight applications if recovery is enabled. To achieve this, the ResourceManager stores its internal state, primarily application-related data and tokens, to the `ResourceManagerStateStore`; the cluster resources are re-constructed when the NodeManagers connect. The available alternatives for the state store are `MemoryResourceManagerStateStore` (a memory-based implementation), `FileSystemResourceManagerStateStore` (file system-based implementation; HDFS can be used for the file system), and `ZKResourceManagerStateStore` (ZooKeeper-based implementation).

Fencing

When running two ResourceManagers, a split-brain situation can arise where both ResourceManagers assume they are active. To avoid this, only a single ResourceManager should be able to perform active operations and the other ResourceManager should be "fenced". The ZooKeeper-based state store (`ZKResourceManagerStateStore`) allows only a single ResourceManager to make changes to the stored state, implicitly fencing the other ResourceManager. This is accomplished by the ResourceManager 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 host such that both ResourceManagers share read-write-admin access, but have exclusive create-delete access. The fencing is implicit and does not 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 ResourceManagers to use different ports (for example, ports on different hosts). To facilitate this, YARN uses the notion of an ResourceManager Identifier (`rm-id`). Each ResourceManager has a unique `rm-id`, and all the RPC configurations (*<rpc-address>*; for example `yarn.resourcemanager.address`) for that ResourceManager can be configured via `<rpc-address>.<rm-id>`. Clients, ApplicationMasters, and NodeManagers use these RPC addresses to talk to the active ResourceManager automatically, even after a failover. To achieve this, they cycle through the list of ResourceManagers in the configuration. This is done automatically and does not require any configuration (as it does in HDFS and MapReduce (MRv1)).

Automatic Failover

By default, ResourceManager HA uses ZKFC (ZooKeeper-based failover controller) for automatic failover in case the active ResourceManager is unreachable or goes down. Internally, the ***StandbyElector*** is used to elect the active ResourceManager. The failover controller runs as part of the ResourceManager (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 ResourceManager to active or standby state, to fail over from one ResourceManager to the other, to get the HA state of an ResourceManager, and to monitor an ResourceManager'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 higher 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 run 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 ResourceManager by default when you enable ResourceManager HA in Cloudera Manager. For more information, including instructions on disabling work preserving recovery, see Work Preserving Recovery for YARN Components on page 403.

> **Note:** ResourceManager HA does not affect the JobHistory Server (JHS). JHS does not 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 runs 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 hosts 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
yarn.resourcemanager.ha.enabled	ResourceManager, NodeManager, Client	false	true	Enable HA
yarn.resourcemanager.ha.rm-ids	ResourceManager, NodeManager, Client	(None)	Cluster-specific, for example: rm1,rm2	Comma-separated list of ResourceManager ids in this cluster.
yarn.resourcemanager.ha.id	ResourceManager	(None)	ResourceManager-specific, for example: 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-ResourceManager RPC) for this ResourceManager. Must be set for all ResourceManagers.
yarn.resourcemanager.scheduler.address.<rm-id>	ResourceManager, Client	(None)	Cluster-specific	The value of yarn.resourcemanager.scheduler.address

Name	Used On	Default Value	Recommended Value	Description
				(AM-ResourceManager RPC) for this ResourceManager. Must beset for all ResourceManagers.
yarn.resourcemanager.admin.address.<rm-id>	ResourceManager, Client/Admin	(None)	Cluster-specific	The value of yarn.resourcemanager.admin.address (ResourceManager administration) for this ResourceManager. Must be set for all ResourceManagers.
yarn.resourcemanager.resource-tracker.address.<rm-id>	ResourceManager, NodeManager	(None)	Cluster-specific	The value of yarn.resourcemanager.resource-tracker.address (NM-ResourceManager RPC) for this ResourceManager. Must be set for all ResourceManagers.
yarn.resourcemanager.webapp.address.<rm-id>	ResourceManager, Client	(None)	Cluster-specific	The value of yarn.resourcemanager.webapp.address (ResourceManager webapp) for this ResourceManager.Must be set for all ResourceManagers.
yarn.resourcemanager.recovery.enabled	ResourceManager	false	true	Enable job recovery on ResourceManager restart or failover.
yarn.resourcemanager.store.class	ResourceManager	org.apache.hadoop.yarn.server.resourcemanager.recovery.FileSystemResourceManagerStateStore	org.apache.hadoop.yarn.server.resourcemanager.recovery.ZKResourceManagerStateStore	The ResourceManagerStateStore implementation to use to store the ResourceManager's internal state. The ZooKeeper- based store supports fencing implicitly. That it, it allows a single ResourceManager to make multiple changes at a time, and hence is recommended.
yarn.resourcemanager.zk-address	ResourceManager	(None)	Cluster-specific	The ZooKeeper quorum to use to store the

Name	Used On	Default Value	Recommended Value	Description
				ResourceManager's internal state.
yarn.resourcemanager.zk-acl	ResourceManager	world:anyone:rwcda	Cluster-specific	The ACLs the ResourceManager uses for the znode structure to store the internal state.
yarn.resourcemanager.zk-state-store.root-node.acl	ResourceManager	(None)	Cluster-specific	The ACLs used for the root host 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 host ACLs are automatically generated on the basis of the ACLs specified through yarn.resourcemanager.zk-acl. 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 403.

Name	Used On	Default Value	Recommended Value	Description
yarn.resourcemanager.ha.automatic-failover.enabled	ResourceManager	true	true	Enable automatic failover
yarn.resourcemanager.ha.automatic-failover.embedded	ResourceManager	true	true	Use the EmbeddedElectorService to pick an active ResourceManager 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 ResourceManager 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.ZKResourceManagerStateStore</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>
    <name>yarn.resourcemanager.work-preserving-recovery.enabled</name>
    <value>true</value>
  </property>

  <!-- ResourceManager1 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>

  <!-- ResourceManager2 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>

<!-- Host 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>
```

Restart the YARN daemons

Start the MapReduce JobHistory server, ResourceManager, and NodeManager on all hosts 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 rmadmin` to Administer ResourceManager HA

You can use `yarn rmadmin` on the command line to manage your ResourceManager HA deployment. `yarn rmadmin` has the following options related to ResourceManager 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 rmadmin`.

Work Preserving Recovery for YARN Components

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

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. You can enable work preserving recovery whether or not you use ResourceManager High Availability.

> **Note:** YARN does not support high availability for the JobHistory Server (JHS). If the JHS goes down, Cloudera Manager will restart it automatically.

> **Note:**
>
> After moving the JobHistory Server to a new host, the URLs listed for the JobHistory Server on the ResourceManager web UI still point to the old JobHistory Server. This affects existing jobs only. New jobs started after the move are not affected. For any existing jobs that have the incorrect JobHistory Server URL, there is no option other than to allow the jobs to roll off the history over time. For new jobs, make sure that all clients have the updated `mapred-site.xml` that references the correct JobHistory Server.

Configuring Work Preserving Recovery Using Cloudera Manager

Enabling Work Preserving Recovery on ResourceManager with Cloudera Manager

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

If you use Cloudera Manager and you enable YARN (MRv2) ResourceManager High Availability on page 396, work preserving recovery is enabled by default for the ResourceManager.

Disabling Work Preserving Recovery on ResourceManager Using Cloudera Manager

To disable Work Preserving Recovery for the ResourceManager:

1. Go to the **YARN** service.
2. Click the **Configuration** tab.
3. Search for `Enable ResourceManager Recovery`.
4. In the **Enable ResourceManager Recovery** field, clear the **ResourceManager Default Group** checkbox.
5. Click **Save Changes**.

Enabling Work Preserving Recovery on NodeManager with Cloudera Manager

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

The default value for the recovery directory is `/var/lib/hadoop-yarn/yarn-nm-recovery.`

Work preserving recovery is enabled by default in Cloudera Manager managed clusters.

These are the steps to enable work preserving recovery for a given NodeManager, if needed:

1. Edit the advanced configuration snippet for `yarn-site.xml` on that NodeManager, and set the value of `yarn.nodemanager.recovery.enabled` to *true*.
2. Configure the directory on the local filesystem where state information is stored when work preserving recovery is enabled.
   a. Go to the **YARN** service.
   b. Click the **Configuration** tab.
   c. Search for `NodeManager Recovery Directory`.
   d. Enter the directory path in the **NodeManager Recovery Directory** field (for example, `/var/lib/hadoop-yarn/yarn-nm-recovery`).
   e. Click **Save Changes**.

Configuring Work Preserving Recovery Using the Command Line

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

Important:

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

After enabling YARN (MRv2) ResourceManager High Availability on page 396, add the following property elements to `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, Set `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 directory under the root partition. If the drive which hosts this directory fails, the NodeManager will also fail. 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 (default is port 0) cannot be used for the NodeManager's RPC server; this could cause the NodeManager to use different ports before and after a restart,

preventing clients from connecting to the NodeManager. The NodeManager RPC address is also important for auxiliary services that run in a YARN cluster.

Auxiliary services should 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 that supports work preserving recovery of the NodeManager.

For more information, see Starting, Stopping, and Restarting Services on page 45.

Example Configuration for Work Preserving Recovery

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

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 higher 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 runs 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 runs 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 372, 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 then decide to revert to the non-HA JobTracker, you 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
`mapred.job.tracker`	`local`	JobTracker, TaskTracker, client	In an HA setup, the logical name of the JobTracker active-standby pair. In a non-HA setup `mapred.job.tracker` is a `host:port` string specifying the JobTracker's RPC address, but in an HA configuration the logical name *must not* include a port number.

Property name	Default	Used on	Description
`mapred.jobtracker.restart.recover`	`false`	JobTracker	Whether to recover jobs that were running in the most recent active JobTracker. Must be set to `true` for JobTracker HA.
`mapred.job.tracker.persist.jobstatus.active`	`false`	JobTracker	Whether to make job status persistent in HDFS. Must be set to `true` for JobTracker HA.
`mapred.job.tracker.persist.jobstatus.hours`	`0`	JobTracker	The number of hours job status information is retained in HDFS. Must be greater than zero for JobTracker HA.
`mapred.job.tracker.persist.jobstatus.dir`	`/jobtracker/jobsInfo`	JobTracker	The HDFS directory in which job status information is kept persistently. The directory must exist and be owned by the `mapred` user.

New configuration parameters

Property name	Default	Used on	Description
`mapred.jobtrackers.<name>`	None	JobTracker, TaskTracker, client	A comma-separated pair of IDs for the active and standby JobTrackers. The `<name>` is the value of `mapred.job.tracker`.
`mapred.jobtracker.rpc-address.<name>.<id>`	None	JobTracker, TaskTracker, client	The RPC address of an individual JobTracker. `<name>` refers to the value of `mapred.job.tracker`; `<id>` refers to one or other of the values in `mapred.jobtrackers.<name>`.
`mapred.job.tracker.http.address.<name>.<id>`	None	JobTracker, TaskTracker	The HTTP address of an individual JobTracker. (In a non-HA setup `mapred.job.tracker.http.address` (with no suffix) is the JobTracker's HTTP address.)
`mapred.ha.jobtracker.rpc-address.<name>.<id>`	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

Property name	Default	Used on	Description
			to mapred.jobtracker.rpc-address.<name>.<id>.
mapred.ha.jobtracker.http-redirect-address.<name>.<id>	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 mapred.job.tracker.http.address.<name>.<id> is needed since the latter may be a wildcard bind address, such as 0.0.0.0:50030, which is not suitable for making requests. Note also that mapred.ha.jobtracker.http-redirect-address.<name>.<id> is the HTTP redirect address for the JobTracker with ID <id> for the pair with the logical name <name> - that is, the address that should be used when that JobTracker is active, and *not* the address that should be redirected to when that JobTracker is the standby.
mapred.ha.jobtracker.id	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 mapred.jobtracker.rpc-address.<name>.<id> properties. It is provided for testing purposes.
mapred.client.failover.proxy.provider.<name>	None	TaskTracker, client	The failover provider class. The only class available is org.apache.hadoop.mapred.ConfiguredFailoverProxyProvider.
mapred.client.failover.max.attempts	15	TaskTracker, client	The maximum number of times to try to fail over.
mapred.client.failover.sleep.base.millis	500	TaskTracker, client	The time to wait before the first failover.
mapred.client.failover.sleep.max.millis	1500	TaskTracker, client	The maximum amount of time to wait between failovers (for exponential backoff).

Property name	Default	Used on	Description
mapred.client.failover.connection.retries	0	TaskTracker, client	The maximum number of times to retry between failovers.
mapred.client.failover.connection.retries.on.timeouts	0	TaskTracker, client	The maximum number of times to retry on timeouts between failovers.
mapred.ha.fencing.methods	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 `mapred.ha.fencing.methods` as `shell(/bin/true)` since the JobTrackers fence themselves, and split-brain is 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>


 <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 `true` 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.

JobTracker High Availability 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

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.

If a Key Trustee Server fails, the following operations are impacted:

- **HDFS Encryption**
  - You cannot create new encryption keys for encryption zones.
  - You can write to and read from existing encryption zones, but you cannot create new zones.
- **Cloudera Navigator Encrypt**
  - You cannot register new Cloudera Navigator Encrypt clients.
  - You can continue reading and writing encrypted data, including creating new mount points, using existing clients.

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 and Clients 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

For new installations, use the **Set up HDFS Data At Rest Encryption** wizard and follow the instructions in Enabling HDFS Encryption Using the Wizard. When prompted, make sure that the **Enable High Availability** option is selected.

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.

After completing the **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:

1. Stop the Key Trustee Server service (**Key Trustee Server service** > **Actions** > **Stop**).
2. Run the **Set Up Key Trustee Server Database** command (**Key Trustee Server service** > **Actions** > **Set Up Key Trustee Server Database**).
3. Run the following command on the Active Key Trustee Server:

```
$ sudo rsync -zcav --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.

4. Run the following command on the Passive Key Trustee Server:

```
$ sudo ktadmin init
```

5. 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.

6. Enable synchronous replication (**Key Trustee Server service** > **Actions** > **Setup Enable Synchronous Replication in HA mode**).
7. Restart the Key Trustee Server service (**Key Trustee Server service** > **Actions** > **Restart**).

For parcel-based Key Trustee Server releases 5.8 and higher, Cloudera Manager automatically backs up Key Trustee Server (using the `ktbackup.sh` script) after adding the Key Trustee Server service. It also schedules automatic backups using `cron`. For package-based installations, you must manually back up Key Trustee Server and configure a `cron` job.

Cloudera Manager configures `cron` to run the backup script hourly. The latest 10 backups are retained in `/var/lib/keytrustee` in cleartext. For information about using the backup script and configuring the `cron` job (including how to encrypt backups), see Backing Up Key Trustee Server and Key Trustee KMS Using the ktbackup.sh Script.

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 -f /var/lib/keytrustee/.ssh/id_rsa
> ```

```
$ sudo ktadmin init --external-address keytrustee01.example.com
$ sudo rsync -zav --exclude .ssl /var/lib/keytrustee/.keytrustee
root@keytrustee02.example.com:/var/lib/keytrustee/.
$ sudo ktadmin db --bootstrap --port 11381 --pg-rootdir /var/lib/keytrustee/db --slave
 keytrustee02.example.com
## For RHEL/CentOS 7, use 'sudo systemctl [stop|start] <service_name>' instead of 'sudo
 service <service_name> [stop|start]' ##
$ sudo service keytrustee-db stop
$ sudo service keytrustee-db start
$ sudo service keytrusteed start
```

Replace *`keytrustee01.example.com`* with the fully qualified domain name (FQDN) of the active Key Trustee Server. Replace *`keytrustee02.example.com`* with the FQDN of the passive Key Trustee Server. Cloudera recommends using the default `/var/lib/keytrustee/db` directory for the PostgreSQL database.

To use a different port for the database, modify the `ktadmin init` and `ktadmin db` commands as follows:

```
$ sudo ktadmin init --external-address keytrustee01.example.com --db-connect
postgresql://localhost:<port>/keytrustee?host=/tmp
$ sudo ktadmin db --bootstrap --port <port> --pg-rootdir /var/lib/keytrustee/db --slave
 keytrustee02.example.com
```

If you use a database directory other than `/var/lib/keytrustee/db`, create or edit the `/etc/sysconfig/keytrustee-db` file and add the following line:

```
ARGS="--pg-rootdir /path/to/db"
```

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 ktadmin init-slave --master keytrustee01.example.com --pg-rootdir
/var/lib/keytrustee/db --no-import-key --no-start
## For RHEL/CentOS 7, use 'sudo systemctl [stop|start] <service_name>' instead of 'sudo
 service <service_name> [stop|start]' ##
$ sudo service keytrustee-db start
$ sudo ktadmin init --external-address keytrustee02.example.com
$ sudo service keytrusteed start
```

Replace *`keytrustee02.example.com`* with the fully qualified domain name (FQDN) of the passive Key Trustee Server. Replace *`keytrustee01.example.com`* with the FQDN of the active Key Trustee Server. Cloudera recommends using the default `/var/lib/keytrustee/db` directory for the PostgreSQL database.

To use a different port for the database, modify the `ktadmin init-slave` command as follows:

```
$ sudo ktadmin init-slave --master keytrustee01.example.com --pg-port <port> --pg-rootdir
 /var/lib/keytrustee/db --no-import-key --no-start
```

If you use a database directory other than `/var/lib/keytrustee/db`, create or edit the `/etc/sysconfig/keytrustee-db` file and add the following line:

```
ARGS="--pg-rootdir /path/to/db"
```

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 `/etc/init.d/keytrustee-db` 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, which protects against key loss.

To enable synchronous replication, run the following command on the active Key Trustee Server:

```
$ sudo 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: Because clients connect to Key Trustee Server using its fully qualified domain name (FQDN), certificates must be issued to the 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 and Clients.

Key Trustee KMS High Availability

CDH 5.4.0 and higher supports Key Trustee KMS high availability. For new installations, you can use the Set up HDFS Data At Rest Encryption wizard to install and configure 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 and Restoring Key Trustee Server and 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 41.
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. Go 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 388.

For details about HA for Impala, see Using Impala through a Proxy for High Availability.

For details about HA for Cloudera Search, see Using Search 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 want 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 *secondary* 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 reflect the most recent updates to the data when updates are made and the 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.

Keeping Replicas Current

The read replica feature includes two different mechanisms for keeping replicas up to date:

Using a Timer

In this mode, replicas are refreshed at a time interval controlled by the configuration option `hbase.regionserver.storefile.refresh.period`. Using a timer is supported in CDH 5.4 and higher.

Using Replication

In this mode, replicas are kept current between a source and sink cluster using HBase replication. This can potentially allow for faster synchronization than using a timer. Each time a flush occurs on the source cluster, a notification is pushed to the sink clusters for the table. To use replication to keep replicas current, you must first set the column family attribute REGION_MEMSTORE_REPLICATION to `false`, then set the HBase configuration property `hbase.region.replica.replication.enabled` to `true`.

> **Important:** Read-replica updates using replication are not supported for the `hbase:meta` table. Columns of hbase:meta must always have their REGION_MEMSTORE_REPLICATION attribute set to `false`.

Enabling Read Replica Support

> **Important:**
>
> Before you enable read-replica support, make sure you 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, you must set several properties.

Table 25: HBase Read Replica Properties

Property Name	Default Value	Description
`hbase.region.replica.replication.enabled`	`false`	The mechanism for refreshing the secondary replicas. If set to `false`, secondary replicas are not guaranteed to be consistent at the row level. Secondary replicas are refreshed at intervals controlled by a timer (`hbase.regionserver.storefile.refresh.period`), and so are guaranteed to be at most that interval of milliseconds behind the primary RegionServer. Secondary replicas read from the HFile in HDFS, and have no access to writes that have not been flushed to the HFile by the primary RegionServer. If `true`, replicas are kept up to date using replication. and the column family has the attribute `REGION_MEMSTORE_REPLICATION` set to `false`, Using replication for read

Property Name	Default Value	Description
		replication of `hbase:meta` is not supported, and `REGION_MEMSTORE_REPLICATION` must always be set to `false` on the column family.
`hbase.regionserver.storefile.refresh.period`	`0` (disabled)	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. 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.

`hbase.master.loadbalancer.class`	`org.apache.hadoop.hbase.master.`
	`balancer.StochasticLoadBalancer`
	(the class name is split for formatting purposes)
		The Java class used for balancing the load of all HBase clients. The default implementation is the `StochasticLoadBalancer`, which is the only load balancer that supports reading data from secondary RegionServers.

`hbase.ipc.client.allowsInterrupt`	`true`	Whether or not to enable interruption of RPC threads at the client. The default value of `true` enables primary RegionServers to access data from other regions' secondary replicas.
`hbase.client.primaryCallTimeout.get`	`10` ms	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.
`hbase.client.primaryCallTimeout.multiget`	`10` ms	The timeout period, in milliseconds, before an HBase client's multi-get request, such as `HTable.get(List<GET>))`, 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.

Configure Read Replicas Using Cloudera Manager

1. Before you can use replication to keep replicas current, you must set the column attribute `REGION_MEMSTORE_REPLICATION` to `false` for the HBase table, using HBase Shell or the client API. See [Activating Read Replicas On a Table](#) on page 426.
2. Select **Clusters** > ***HBase***.
3. Click the **Configuration** tab.
4. Select **Scope** > **HBase or HBase Service-Wide**.
5. Select **Category** > **Advanced**.

6. 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.
7. Using the same XML syntax as Configure Read Replicas Using the Command Line on page 425 and the chart above, create a configuration and paste it into the text field.
8. Click **Save Changes** to commit the changes.

Configure Read Replicas Using the Command Line

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

1. Before you can use replication to keep replicas current, you must set the column attribute `REGION_MEMSTORE_REPLICATION` to `false` for the HBase table, using HBase Shell or the client API. See Activating Read Replicas On a Table on page 426.
2. Add the properties from Table 25: HBase Read Replica Properties on page 423 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>
```

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

Configuring Rack Awareness for Read Replicas

Rack awareness for read replicas is modeled after the mechanism used for rack awareness in Hadoop. Its purpose is to ensure that some replicas are on a different rack than the RegionServer servicing the table. The default implementation, which you can override by setting `hbase.util.ip.to.rack.determiner`, to custom implementation, is `ScriptBasedMapping`, which uses a *topology map* and a *topology script* to enforce distribution of the replicas across racks.

Creating a Topology Map

The topology map assigns hosts to racks. It is read by the topology script. A rack is a logical grouping, and does not necessarily correspond to physical hardware or location. Racks can be nested. If a host is not in the topology map, it is assumed to be a member of the default rack. The following map uses a nested structure, with two data centers which each have two racks. All services on a host that are rack-aware will be affected by the rack settings for the host.

If you use Cloudera Manager, do not create the map manually. Instead, go to **Hosts**, select the hosts to assign to a rack, and select **Actions for Selected** > **Assign Rack**.

```
<topology>
  <node name="host1.example.com" rack="/dc1/r1"/>
  <node name="host2.example.com" rack="/dc1/r1"/>
  <node name="host3.example.com" rack="/dc1/r2"/>
  <node name="host4.example.com" rack="/dc1/r2"/>
  <node name="host5.example.com" rack="/dc2/r1"/>
  <node name="host6.example.com" rack="/dc2/r1"/>
  <node name="host7.example.com" rack="/dc2/r2"/>
  <node name="host8.example.com" rack="/dc2/r2"/>
</topology>
```

Creating a Topology Script

The topology script determines rack topology using the topology map. By default, CDH uses `/etc/hadoop/conf.cloudera.YARN-1/topology.py` To use a different script, set `net.topology.script.file.name` to the absolute path of the topology script.

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 if a metastore becomes unavailable.

Prerequisites

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

Limitations

Sentry HDFS synchronization does not support Hive metastore HA.

Enabling Hive Metastore High Availability Using Cloudera Manager

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

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

   If more than one role group applies to this configuration, edit the value for the appropriate role group. See Modifying Configuration Properties Using Cloudera Manager on page 13.

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

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

Enabling Hive Metastore High Availability Using the Command Line

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

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

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

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

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

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

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

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

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

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

      `hive --service metatool -listFSRoot`

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

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

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

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

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

```
$ sudo service hive-metastore stop
$ /usr/lib/hive/bin/beeline
beeline> !connect jdbc:hive2://metastore2.example.com:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000> SHOW TABLES;
show tables;
+-----------+
| tab_name  |
+-----------+
+-----------+
No rows selected (0.238 seconds)
0: jdbc:hive2://localhost:10000>
```

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

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

Configuring HiveServer2 High Availability in CDH

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

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

Enabling HiveServer2 High Availability Using Cloudera Manager

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

1. Add multiple HiveServer2 instances to your cluster:
   1. Go to the **Hive** service.
   2. Click the **Instances** tab, and then click **Add Role Instances**.
   3. On the **Add Role Instances to Hive** page under the **HiveServer2** column heading, click **Select hosts**, and select the hosts that should have a HiveServer2 instance.
   4. Click **OK**, and then click **Continue**. You will be brought to the Instances page where you can start the new HiveServer2 instances.
2. Click the **Configuration** tab.
3. Select **Scope** > **HiveServer2**.
4. Select **Category** > **Main**.

5. Locate the **HiveServer2 Load Balancer** property or search for it by typing its name in the Search box.
6. Enter values for *<hostname>*:*<port number>*. For example, hs2load_balancer.example.com:10015.

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

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

Configuring HiveServer2 to Load Balance Behind a Proxy on Unmanaged Clusters

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

Unmanaged Clusters with Kerberos Enabled

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

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

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

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

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

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

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

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

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

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

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

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

4. Validate the keytab by running `klist`:

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

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

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

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

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

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

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

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

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

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

Unmanaged Clusters WITHOUT Kerberos

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

1. Download load-balancing proxy software of your choice on a single host. For example, see [Example HAProxy Configuration](#).
2. Configure the software, typically by editing a configuration file. Usually this configuration includes:
   a. Setting the port for the load balancer to listen on and to relay HiveServer2 requests back and forth.
   b. Setting the port and hostname for each HiveServer2 host. These are the hosts from which the load balancer chooses when relaying each query.
3. Run the load-balancing proxy server and point it at the configuration file.
4. Point all scripts, jobs, or application configurations to the new proxy server instead of any specific HiveServer2 instance.

Example HAProxy Configuration

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

1. Download the appropriate [HAProxy software](#).
2. As the `root` user, install HAProxy:

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

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

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

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

#---------------------------------------------------------------------
# common defaults that all the 'listen' and 'backend' sections will
```

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

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

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

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

4. Set HAProxy to start when the system starts:

```
chkconfig haproxy on
```

5. Start HAProxy:

```
service haproxy start
```

Hue High Availability

This page explains how to configure Hue for high availability with Cloudera Manager and at the command line. It assumes that you have the Hue service installed and one or more Hue server roles defined. If not, see Adding a Hue Service and Role Instance on page 222.

> **Important: Cloudera strongly recommends an external database for clusters with multiple Hue servers (or "hue" users).** With the default embedded database (one per server), in a multi-server environment, the data on server "A" *appears* lost when working on server "B" and vice versa. Use an external database, and configure each server to point to it to ensure that no matter which server is being used by Hue, your data is always accessible.

Configuring a Cluster for Hue High Availability Using Cloudera Manager

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

In Cloudera Manager, you can configure your Hue cluster for high availability by adding a Hue load balancer and multiple Hue servers.

Prerequisite

- An external database was added and each Hue server is configured to use it. See Connect Hue to an External Database.

Configuring Hue for High Availability in Cloudera Manager

1. Go to the Hue service.
2. Select the Hue **Instances** tab.
3. Add one or more Hue servers to an existing Hue server role. At least two Hue server roles are required for high availability:
    a. Click **Add Role Instances**.
    b. Click **Select hosts** under Hue Server (HS).
    c. Check the box for each host on which you want a Hue server (which adds **HS** icons).
    d. Click **OK**.
4. Add one load balancer:
    a. Click **Select hosts** under Load Balancer (LB).
    b. Check the box for the host on which you want the load balancer (which adds an **LB** icon).
    c. Click **OK**.
5. Click **Continue**.
6. Select the **Configuration** tab and filter on **Scope** > **Hue Server** and **Category** > **Security**.
7. Find the property, **Enable TLS/SSL for Hue**, and check the box by **Hue Server Default Group**.
8. Clear Scope and Category filters and review other options to ensure they meet your needs. Some to consider are:
    - **Hue Load Balancer Port** - The Apache Load Balancer listens on this port. The default is 8889.
    - **Path to TLS/SSL Certificate File** - The TLS/SSL certificate file.
    - **Path to TLS/SSL Private Key File** - The TLS/SSL private key file.
9. Save any configuration changes.
10. Start the load balancer and new Hue servers:
    a. Check the box by each new role type.
    b. Select **Actions for Selected** > **Start**.
    c. Click **Start** and **Close**.

For more information, see Automatic High Availability and Load Balancing of Hue.

Configuring a Cluster for Hue High Availability Using the Command Line

This section applies to unmanaged deployments *without* Cloudera Manager. It explains how to install and configure **nginx** from the command line. To make the Hue service highly available, configure at least two instances of the Hue service, each on a different host. Also configure the **nginx** load balancer to direct traffic to an alternate host if the primary host becomes unavailable. For advanced configurations, see the nginx documentation.

Prerequisites

- The Hue service is installed and **two** or more Hue server roles are configured.
- You have network access through SSH to the host machines with the Hue server role(s).
- An external database was added each Hue server is configured to use it. See Connect Hue to an External Database.

Installing and Configuring 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. With SSH, log in as the **root** user to the host machine of one of the Hue instances.
2. Install nginx:

   Red Hat/Centos:

```
yum install nginx
```

   Debian/Ubuntu:

```
apt-get install nginx
```

3. Create the following Hue cluster configuration file:

```
/etc/nginx/conf.d/hue.conf
```

4. Configure `hue.conf` with the following template:

```
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;

    client_max_body_size 0;
    location / {
        proxy_pass http://hue;
        proxy_set_header Host $http_host;
        proxy_set_header X-Forwarded-For $remote_addr;

        # 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 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 TLS/SSL:
  - Uncomment these lines, and replace the path names with the paths 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. Start nginx:

```
sudo service nginx start
```

7. Test your installation by opening the Hue application in a web browser, using the following URL:
   - Without TLS/SSL: `http://`*NGINX_HOSTNAME*`:8001`
   - With TLS/SSL: `https://`*NGINX_HOSTNAME*`:8001`

   NGINX_HOSTNAME is the name of the host machine where you installed nginx.

   > **Note:** To check if Hue is still running, use `http://<HUE_HOST>/desktop/debug/is_alive`. A status code of `200` 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.
9. If necessary, configure your backend database for high availability. Consult the vendor documentation for the database that you configured for Hue.

Oozie 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 for Oozie High Availability

- 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.

To enable Kerberos authentication, see Enabling Kerberos Authentication Using the Wizard.

For information on setting up TLS/SSL communication with Oozie HA enabled, see Additional Considerations when Configuring TLS/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. Enter the FQDN and port number of the Oozie load balancer. For example:

```
load-bal.example.com:12345
```

6. 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. In addition, Cloudera Manager generates Kerberos credentials for the new Oozie server and regenerates credentials for existing servers.

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.

To enable Kerberos authentication for an Oozie HA-enabled deployment, see Configuring Oozie HA with Kerberos.

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 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:

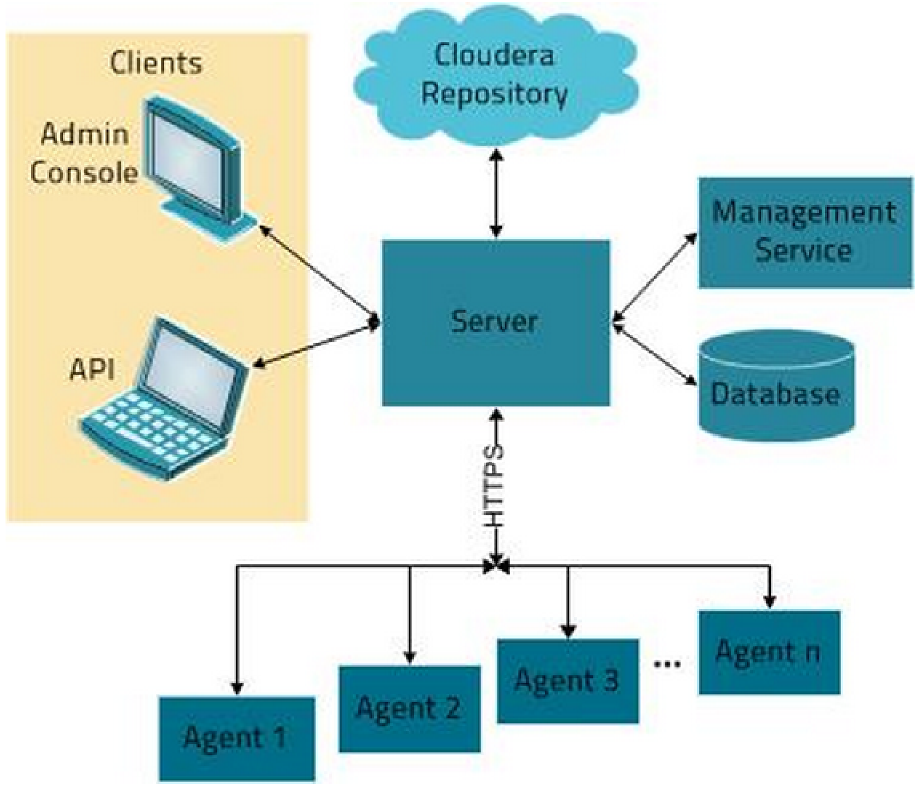


Figure 13: Cloudera Manager Architecture

- 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 TLS/SSL 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 470.

Cloudera Manager Failover Protection

A CDH cluster managed by Cloudera Manager can have only one instance of Cloudera Manager active at a time. A Cloudera Manager instance is backed by a single database instance that stores configurations and other operational data.

CDH deployments that use highly available configurations for Cloudera Manager can configure a “standby” instance of Cloudera Manager that takes over automatically if the primary instance fails. In some situations, a second instance of Cloudera Manager may become active during maintenance or upgrade activities or due to operator error. If two instances of Cloudera Manager are active at the same time and attempt to access the same database, data corruption can result, making Cloudera Manager unable to manage the cluster.

In Cloudera Manager 5.7 and higher, Cloudera Manager automatically detects when more than one instance of Cloudera Manager is running and logs a message in the `/var/log/cloudera-scm-server/cloudera-scm-server.log` file. For example:

```
2016-02-17 09:47:27,915 WARN
main:com.cloudera.server.cmf.components.ScmActive:
ScmActive detected spurious CM :
hostname=sysadmin-scm-2.mycompany.com/172.28.197.136,bootup true
2016-02-17 09:47:27,916 WARN
main:com.cloudera.server.cmf.components.ScmActive: ScmActive:
The database is owned by sysadmin-scm-1.mycompany.com/172.28.197.242
2016-02-17 09:47:27,917 ERROR
main:com.cloudera.server.cmf.bootstrap.EntityManagerFactoryBean: ScmActiveat bootup:
The configured database is being used by another instance of Cloudera Manager.
```

In addition, the second instance of Cloudera Manager is automatically shut down, resulting in messages similar to the following in the log file:

```
2016-02-17 09:47:27,919 ERROR main:com.cloudera.server.cmf.Main: Serverfailed.2016-02-17
 09:47:27,919
ERROR main:com.cloudera.server.cmf.Main:
Serverfailed.org.springframework.beans.factory.BeanCreationException:
Error creatingbean with name 'com.cloudera.server.cmf.TrialState':
Cannot resolvereference to bean 'entityManagerFactoryBean' while setting
constructorargument;
nested exception isorg.springframework.beans.factory.BeanCreationException:
Error creatingbean with name 'entityManagerFactoryBean':
FactoryBean threw exception onobject creation; nested exception is
java.lang.RuntimeException: ScmActiveat bootup:
Failed to validate the identity of Cloudera Manager.
```

When a Cloudera Manager instance fails or becomes unavailable and remains offline for more than 30 seconds, any new instance that is deployed claims ownership of the database and continues to manage the cluster normally.

Disabling Automatic Failover Protection

You can disable automatic shutdown by setting a Java option and restarting Cloudera Manager:

1. On the host where Cloudera Manager server is running, open the following file in a text editor:

```
/etc/default/cloudera-scm-server
```

2. Add the following property (separate each property with a space) to the line that begins with `export CMF_JAVA_OPTS`:

```
-Dcom.cloudera.server.cmf.components.scmActive.killOnError=false
```

   For example:

```
export CMF_JAVA_OPTS="-Xmx2G -XX:MaxPermSize=256m -XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/tmp -Dcom.cloudera.server.cmf.components.scmActive.killOnError=false”
```

3. Restart the Cloudera Manager server by running the following command on the Cloudera Manager server host:

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

> **Note:** When you disable automatic shutdown, a message is still logged when more than one instance of Cloudera Manager is running .

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

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 469.

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.

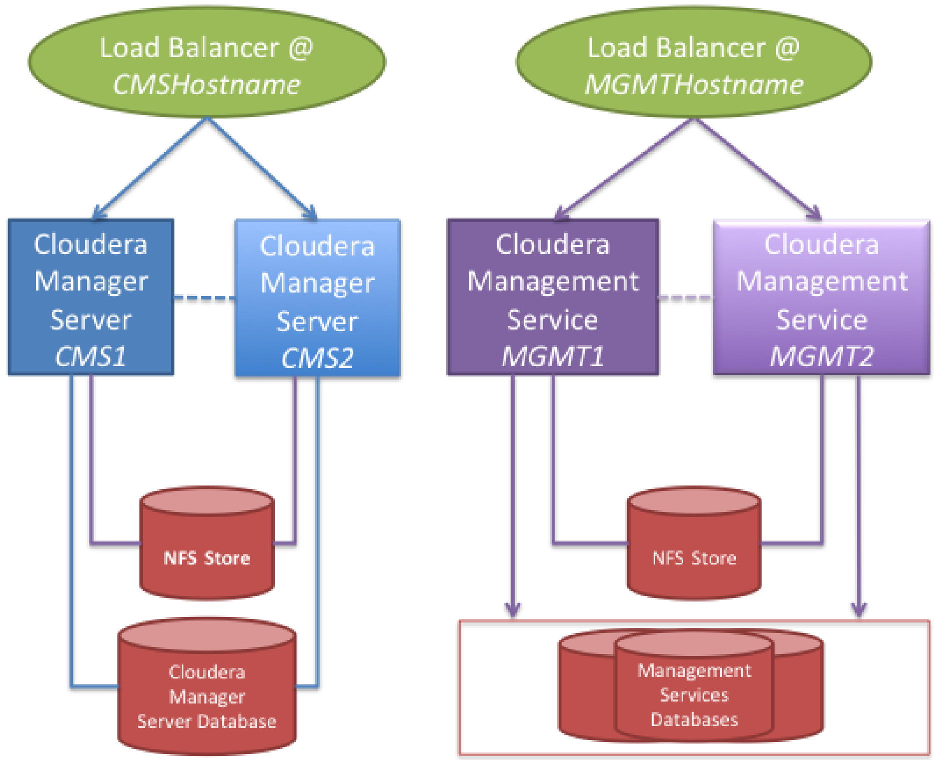


Figure 14: High-level layout of components for Cloudera Manager high availability

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.
>
> HAProxy 1.5.4-2 has a bug that affects the functioning of tcp-check. Cloudera recommends that you use version 1.6.3.

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 CDH and Cloudera Manager 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 *MGMTHostname*

```
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 Datastores.

   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 469 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 452, which describes how to set up the Cloudera Management Service.

See:

- Installation Path B - Installation Using Cloudera Manager Parcels or 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. Clear 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 470.

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 - Installation Using Cloudera Manager Parcels or 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 470 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-service-monitor 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-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 - Installation Using Cloudera Manager Parcels or Packages).

   b. Install the Oracle Java JDK version that is required for your deployment, if it is not already installed on the host. See CDH and Cloudera Manager 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.

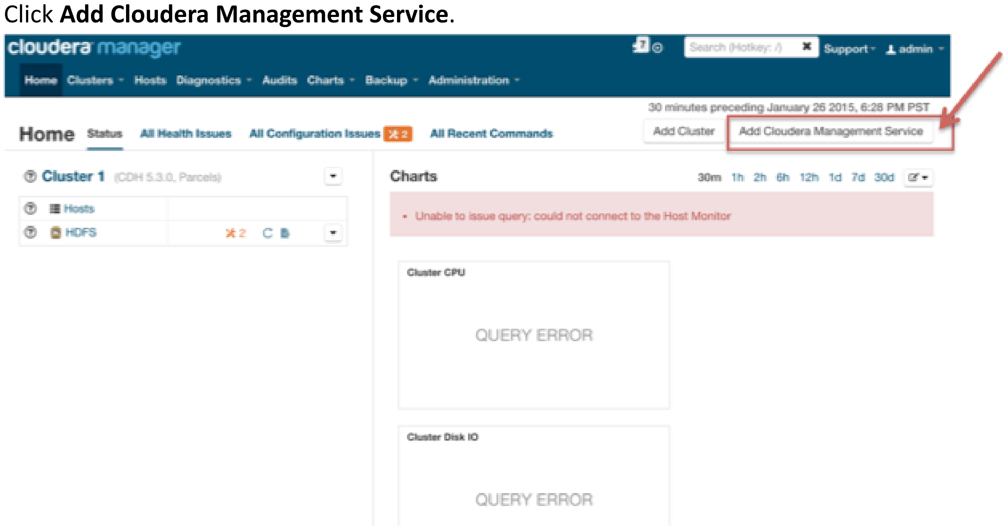
e. Restart the agent on *`MGMT1`* (this also starts the agent if it is not running):

```
$ 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.)

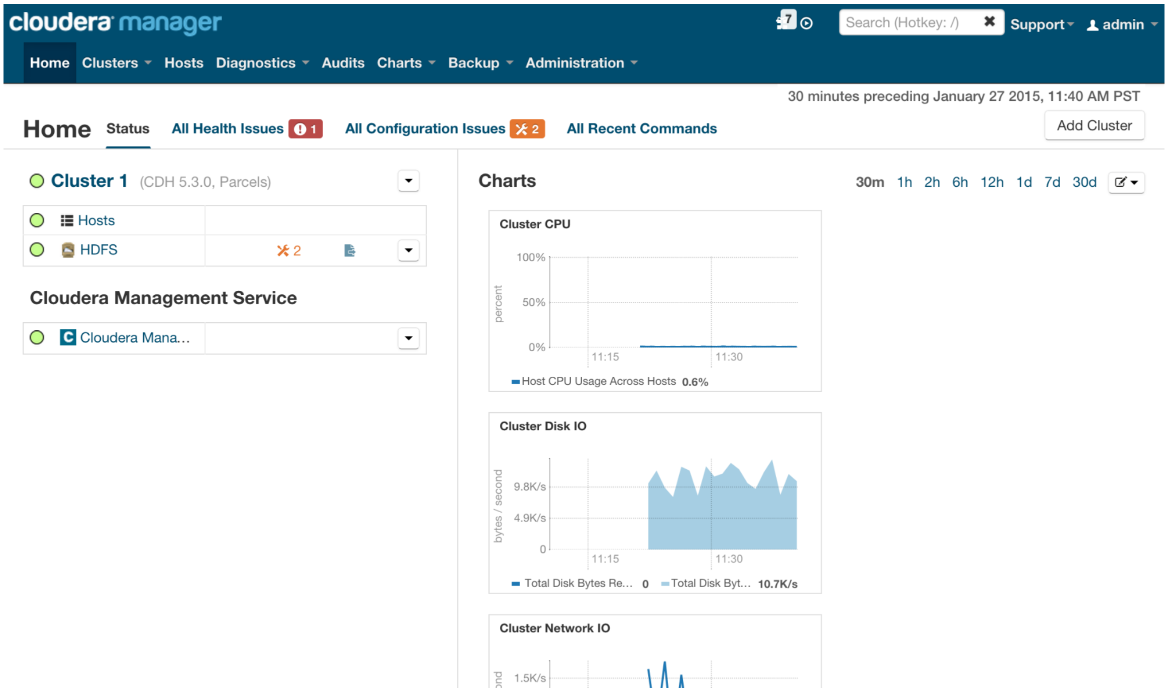
b. Click **Add Cloudera Management Service**.



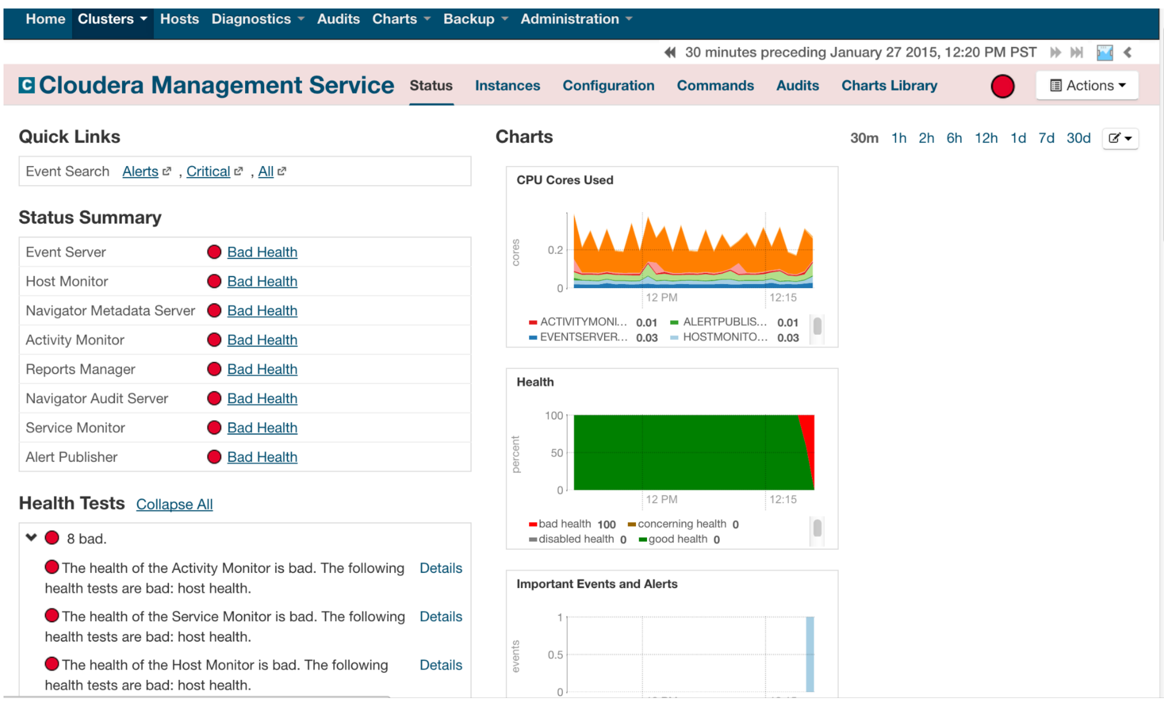
g. Make sure you install all of the roles of the Cloudera Management Service on the host named *`MGMTHostname`*.

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:



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:

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 TLS/SSL, see TLS and Kerberos Configuration for Cloudera Manager High Availability on page 470 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** > **Status** tab, 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 - Installation Using Cloudera Manager Parcels or Packages.

   b. Install the Oracle Java JDK version that is required for your deployment, if it is not already installed on the host. See CDH and Cloudera Manager 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 454.

   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 463).
- 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* andf 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*:

      - RHEL-compatible 7 and higher:

```
#!/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 next_stop_hard
    service cloudera-scm-agent stop
    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
```

- All other Linux distributions:

```
#!/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

Cloudera Manager supports TLS for encrypted network communications, and it supports integration with Kerberos for authentication (see Configuring TLS Security for Cloudera Manager and Configuring Authentication in Cloudera Manager for details). Configuring TLS- or Kerberos-enabled Cloudera Manager clusters for high availability requires the additional steps discussed below:

- TLS Considerations for Cloudera Manager High Availability
  - Configure Load Balancers for TLS Pass-Through
  - Server Certificate Requirements for HA Deployments
  - Cloudera Manager Agent Host Requirements for HA Deployments
- Kerberos Considerations for Cloudera Manager High Availability
  - Server Configuration Requirements for HA
  - Re-Generate Kerberos Credentials

Example hostnames used throughout Configuring Cloudera Manager for High Availability With a Load Balancer on page 439 are summarized in the table below.

Host description	Example hostnames
Load balancer for Cloudera Manager Server	CMSHostname
Cloudera Manager Server (primary)	CMS1
Cloudera Manager Server (secondary)	CMS2
Load balancer for Cloudera Management Service	MGMTHostname
Cloudera Management Service (primary)	MGMT1
Cloudera Management Service (secondary)	MGMT2

TLS Considerations for Cloudera Manager High Availability

When successfully configured for high availability and for TLS, the Cloudera Manager Admin Console is accessed using the host name or IP address of the load balancer:

```
https://[CMSHostname]:7183
```

This assumes that the load balancer has been set up for TLS pass-through and that the Cloudera Manager Server host has been set up as detailed below.

Configure Load Balancers for TLS Pass-Through

As detailed in Configuring Cloudera Manager for High Availability With a Load Balancer on page 439, high availability for Cloudera Manager Server clusters requires secondary nodes that act as backups for the primary Cloudera Manager Server and Cloudera Management Service nodes, respectively. Only the primary nodes are active at any time, but if these fail, requests are redirected by a load balancer (CMSHostname or MGMTHostname) to the appropriate secondary node.

When the Cloudera Manager Server cluster is configured for TLS in addition to high availability, the load balancers must be configured for TLS *pass-through*—traffic from clients is not decrypted until it receives the actual server host system. Keep this in mind when you are Setting up the Load Balancer on page 445 for your Cloudera Manager High Availability deployment.

Server Certificate Requirements for HA Deployments

TLS-enabled Cloudera Manager Server clusters require certificates that authenticate the host identity prior to encryption, as detailed in Level 2: Configuring TLS Verification of Cloudera Manager Server by the Agents. When deploying Cloudera Manager Server for high availability, however, the certificate must identify the load balancer and both primary and secondary nodes (rather than the primary host alone). That means you must create your certificate signing request (CSR) as follows:

- Use the FQDN of the load balancer (for example, (*CMSHostname*) for the CN (common name).
- Use the primary and secondary Cloudera Manager Server host names (for example, *CMS1* and *CMS2*, respectively) for the SubjectAlternativeName (SAN) values.

To create a CSR using these example load balancer and server host names:

```
keytool -genkeypair -alias loadBalProxyCMS  -keyalg RSA -keystore keystoreName.jks \
-keysize 2048 -dname "CN=CMSHostname, OU=Department, O=Example, \
L=City, ST=State, C=US" -storepass password \
-keypass password

keytool -certreq -alias loadBalProxyCMS -keystore keystoreName.jks -file
sigRequestHA_LB_1.csr \
-storepass password -keypass password  -ext san=dns:CMS1,dns:CMS2
```

Alternatively, if the Cloudera Manager Server certificates on the hosts do not specify the load balancer name and SAN names, you can make the following change to the configuration. From the Cloudera Manager Admin Console, go to:

- **Administration** > **Ports and Addresses**
- Enter the FQDN of the load balancer in the **Cloudera Manager Hostname Override**

In addition to using correctly created certificates (or over-riding the hostname), you must:

- Store the keystore and truststore in the same path on both primary and secondary Cloudera Manager Server hosts (*CMS1*, *CMS2*), or point to the same shared network mount point from each host.

Cloudera Manager Agent Host Requirements for HA Deployments

Cloudera Manager Server hosts can present their certificate to agents prior to encrypting the connection (TLS Level 2: see Enabling Server Certificate Verification on Cloudera Manager Agents for details). For a high availability deployment:

- Use the same setting for `verify_cert_file` (in the `/etc/cloudera-scm-agent/config.ini` file) on each agent host system. To simplify the set, share the file path to `verify_cert_file` or copy the files manually as specified in the config file between *MGMT1* and *MGMT2*.

Cloudera Manager Agent hosts can present certificates to requesting processes such as the Cloudera Manager Server prior to encryption (TLS Level 3: seeConfiguring Agent Certificate Authentication). For a high availability deployment:

- Share the certificate and key for use by all Cloudera Manager Agent host systems on NFS, or copy them to the same path on both *MGMT1* and *MGMT2*.

> **Important:** Restart `cloudera-scm-agent` after making changes to the certificates or other files, or to the configuration.

Kerberos Considerations for Cloudera Manager High Availability

As detailed in Creating Hosts for Primary and Secondary Servers on page 444, primary and secondary nodes that comprise a Cloudera Manager High Availability cluster must be configured the same (only one host is active at any given time). That means that if the cluster uses Kerberos for authentication, the Kerberos configuration on the primary and secondary nodes must also be the same.

Server Configuration Requirements for HA

When configuring high availability for Kerberos-enabled Cloudera Manager clusters, you must:

- Install Kerberos client libraries in the same path on both primary (for example, *CMS1*) and secondary (*CMS2*) Cloudera Manager Server hosts .
- Configure the `/etc/krb5.conf` file identically across the Cloudera Manager Server and Cloudera Management Service hosts (*CMS1*, *CMS2*, *MGMT1*, *MGMT2*).
- If the Cloudera Manager Server primary host (*CMS1*) is configured to store the Cloudera Manager Server KDC access credentials in `/etc/cloudera-scm-server`, use this same path on the secondary host (*CMS2*).

Re-Generate Kerberos Credentials

Configuring the Cloudera Management Service for high availability using an existing Cloudera Manager Server cluster (as discussed in Installing the Primary on page 454) results in the Cloudera Management Service not starting, as shown here:

This is expected. To resolve, re-generate the Kerberos credentials for the roles:

1. Log in to the Cloudera Manager Admin Console.
2. Select **Administration** > **Kerberos** > **Credentials** > **Generate Credentials**.

Backup and Disaster Recovery

Cloudera Manager provides an integrated, easy-to-use management solution for enabling data protection on the Hadoop platform. Cloudera Manager enables you to replicate data across datacenters for disaster recovery scenarios. Replications can include data stored in HDFS, data stored in Hive tables, Hive metastore data, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore. When critical data is stored on HDFS, Cloudera Manager helps to ensure that the data is available at all times, even in case of complete shutdown of a datacenter.

You can also replicate HDFS data to and from Amazon S3.

You can also use the HBase shell to replicate HBase data. (Cloudera Manager does not manage HBase replications.)

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 583.

You can also use Cloudera Manager to schedule, save, and restore snapshots of HDFS directories and HBase tables.

Cloudera Manager provides key functionality in the Cloudera Manager Admin Console:

- **Select** - Choose datasets that are critical for your business operations.
- **Schedule** - Create an appropriate schedule for data replication and snapshots. Trigger replication and snapshots as required 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 quickly.

Replication works seamlessly across Hive and HDFS—you can set it up on files or directories in HDFS and on tables in Hive—without manual translation of Hive datasets to HDFS datasets, or vice versa. Hive metastore information is also replicated, so applications that depend on table definitions stored in Hive will work correctly on both the replica side and the source side as table definitions are updated.

Replication is built on a hardened version of `distcp`. It uses the scalability and availability of MapReduce and YARN to copy files in parallel, using a specialized MapReduce job or YARN application that runs diffs and transfers only changed files from each mapper to the replica side. Files are selected for copying based on their size and checksums.

You can also perform a “dry run” to verify configuration and understand the cost of the overall operation before actually copying the entire dataset.

Port Requirements for Backup and Disaster Recovery

Make sure that the following ports are open and accessible on the source hosts to allow communication between the source and destination Cloudera Manager servers and the HDFS, Hive, MapReduce, and YARN hosts:

Table 26:

Service	Default Port
Cloudera Manager Admin Console HTTP	7180
Cloudera Manager Admin Console HTTPS (with TLS enabled)	7183
Cloudera Manager Agent	9000
HDFS NameNode	8020

Service	Default Port
Key Management Server (KMS)	16000
HDFS DataNode	50010
WebHDFS	50070
YARN Resource Manager	8032

See Ports for more information, including how to verify the current values for these ports.

Data Replication

Cloudera Manager enables you to replicate data across datacenters for disaster recovery scenarios. Replications can include data stored in HDFS, data stored in Hive tables, Hive metastore data, and Impala metadata (catalog server metadata) associated with Impala tables registered in the Hive metastore. When critical data is stored on HDFS, Cloudera Manager helps to ensure that the data is available at all times, even in case of complete shutdown of a datacenter.

You can also replicate HDFS data to and from Amazon S3 and you can replicate Hive data and metadata to and from Amazon S3.

For an overview of data replication, view this video about Backing Up Data Using Cloudera Manager.

You can also use the HBase shell to replicate HBase data. (Cloudera Manager does not manage HBase replications.)

For recommendations on using data replication and Sentry authorization, see Configuring Sentry to Enable BDR Replication.

Cloudera License Requirements for Replication

Both the *source* and *destination* clusters must have a Cloudera Enterprise license.

Supported and Unsupported Replication Scenarios

Supported Replication Scenarios

In Cloudera Manager 5, replication is supported between CDH 5 or CDH 4 clusters. The following tables describe support for HDFS and Hive replication.

Service	Source			Destination		
	Cloudera Manager Version	CDH Version	Comment	Cloudera Manager Version	CDH Version	Comment
HDFS, Hive	4	4		5	4	
HDFS, Hive	4	4.4 or higher		5	5	
HDFS, Hive	4 or 5	5	SSL enabled on Hadoop services	4 or 5	5	SSL enabled on Hadoop services
HDFS, Hive	4 or 5	5	SSL enabled on Hadoop services	4 or 5	5	SSL *not* enabled on Hadoop services
HDFS, Hive	4 or 5	5.1	SSL enabled on Hadoop services and YARN	4 or 5	4 or 5	
HDFS, Hive	5	4		4.7.3 or higher	4	

Service	Source			Destination		
	Cloudera Manager Version	CDH Version	Comment	Cloudera Manager Version	CDH Version	Comment
HDFS, Hive	5	4		5	4	
HDFS, Hive	5	5		5	5	
HDFS, Hive	5	5		5	4.4 or higher	
HDFS, Hive	5.7	5.7, with Isilon storage		5.7	5.7, with Isilon storage	
HDFS, Hive	5.7	5.7	See Supported Replication Scenarios for Clusters using Isilon Storage.	5.7	5.7, with Isilon storage	See Supported Replication Scenarios for Clusters using Isilon Storage.
HDFS, Hive	5.7	5.7, with Isilon storage		5.7	5.7	
HDFS, Hive	5.8	5.8, with or without Isilon Storage	• Kerberos can be enabled • See Supported Replication Scenarios for Clusters using Isilon Storage.	5.8	5.8, with or without Isilon Storage	• Kerberos can be enabled • See Supported Replication Scenarios for Clusters using Isilon Storage.

Unsupported Replication Scenarios

Service	Source			Destination		
	Cloudera Manager Version	CDH Version	Comment	Cloudera Manager Version	CDH Version	Comment
Any	4 or 5	4 or 5	Kerberos enabled.	4 or 5	4 or 5	Kerberos not enabled
Any	4 or 5	4 or 5	Kerberos not enabled.	4 or 5	4 or 5	Kerberos enabled
HDFS, Hive	4 or 5	4	Where the replicated data includes a directory that contains a large number of files or subdirectories (several hundred thousand entries), causing out-of-memory errors. To work around this issue, follow this procedure.	4 or 5	5	
Hive	4 or 5	4	**Replicate HDFS Files** is disabled.	4 or 5	4 or 5	Over-the-wire encryption is enabled.

Service	Source			Destination		
	Cloudera Manager Version	**CDH Version**	**Comment**	**Cloudera Manager Version**	**CDH Version**	**Comment**
Hive	4 or 5	4	Replication can fail if the NameNode fails over during replication.	4 or 5	5, with high availability enabled	Replication can fail if the NameNode fails over during replication.
Hive	4 or 5	4	The clusters use different Kerberos realms.	4 or 5	5	An older JDK is deployed. (Upgrade the CDH 4 cluster to use JDK 7 or JDK6u34 to work around this issue.)
Any	4 or 5	4	SSL enabled on Hadoop services.	4 or 5	4 or 5	
Hive	4 or 5	4.2 or higher	If the Hive schema contain views.	4 or 5	4	
HDFS	4 or 5	4, with high availability enabled	Replications fail if NameNode failover occurs during replication.	4 or 5	5, without high availability	Replications fail if NameNode failover occurs during replication.
HDFS	4 or 5	4 or 5	Over the wire encryption is enabled.	4 or 5	4	
HDFS	4 or 5	5	Clusters where there are URL-encoding characters such as % in file and directory names.	4 or 5	4	
Hive	4 or 5	4 or 5	Over the wire encryption is enabled and **Replicate HDFS Files** is enabled.	4 or 5	4	
Hive	4 or 5	4 or 5	From one cluster to the same cluster.	4 or 5	4 or 5	From one cluster to the same cluster.
HDFS, Hive	4 or 5	5	Where the replicated data includes a directory that contains a large number of files or subdirectories (several hundred thousand entries), causing out-of-memory errors. To work around this issue, follow this procedure.	4 or 5	4	
HDFS	4 or 5	5	The clusters use different Kerberos realms.	4 or 5	4	An older JDK is deployed. (Upgrade the CDH 4 cluster to use JDK 7 or JDK6u34 to work around this issue.)
Hive	4 or 5	5	**Replicate HDFS Files** is enabled and the clusters use different Kerberos realms.	4 or 5	4	An older JDK is deployed. (Upgrade the CDH 4 cluster to use JDK 7 or JDK6u34 to work around this issue.)

Service	Source			Destination		
	Cloudera Manager Version	CDH Version	Comment	Cloudera Manager Version	CDH Version	Comment
Any	4 or 5	5	SSL enabled on Hadoop services and YARN.	4 or 5	4 or 5	
Any	4 or 5	5	SSL enabled on Hadoop services.	4 or 5	4	
HDFS	4 or 5	5, with high availability enabled	Replications fail if NameNode failover occurs during replication.	4 or 5	4, without high availability	Replications fail if NameNode failover occurs during replication.
HDFS, Hive	5	5		4	4	
Hive	5.2	5.2 or lower	Replication of Impala UDFs is skipped.	4 or 5	4 or 5	

Workaround for replicated data that includes a directory that contains several hundred thousand files or subdirectories:

1. On the destination Cloudera Manager instance, go to the HDFS service page.
2. Click the **Configuration** tab.
3. Select **Scope** > ***HDFS service name*** **(Service-Wide)** and **Category** > **Advanced**.
4. Locate the **HDFS Replication Advanced Configuration Snippet** property.
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.
6. Click **Save Changes** to commit the changes.

HDFS Replication to Amazon S3

You can use Cloudera Manager to replicate HDFS data to and from Amazon S3, however you cannot replicate data from one Amazon S3 instance to another using Cloudera Manager. You must have the appropriate credentials to access the Amazon S3 account and you must create buckets in Amazon S3 to store the replicated files.

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

To configure HDFS replication to Amazon S3:

1. Create an **External Account** using Amazon S3 credentials. See Configuring an External Account for Amazon S3 Replication on page 478.
2. Create an **HDFS Replication Schedule**. See HDFS Replication on page 479.

Supported Replication Scenarios for Clusters using Isilon Storage

Note the following when scheduling replication jobs for clusters that use Isilon storage:

- As of CDH 5.8 and higher, Replication is supported for clusters using Kerberos and Isilon storage on the source or destination cluster, or both. See Configuring Replication with Kerberos and Isilon on page 206. Replication between clusters using Isilon storage and Kerberos is not supported in CDH 5.7.
- Make sure that the `hdfs` user is a superuser in the Isilon system. If you specify alternate users with the **Run As** option when creating replication schedules, those users must also be superusers.
- Cloudera recommends that you use the Isilon `root` user for replication jobs. (Specify `root` in the **Run As** field when creating replication schedules.)

- Select the **Skip checksum checks** property when creating replication schedules.
- Clusters that use Isilon storage do not support snapshots. Snapshots are used to ensure data consistency during replications in scenarios where the source files are being modified. Therefore, when replicating from an Isilon cluster, Cloudera recommends that you do not replicate Hive tables or HDFS files that could be modified before the replication completes.

See Using CDH with Isilon Storage on page 199.

Designating a Replication Source

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

The Cloudera Manager Server that you are logged into is the destination for replications set up using that Cloudera Manager instance. From the Admin Console of this destination Cloudera Manager instance, you can designate a peer Cloudera Manager Server as a source of HDFS and Apache Hive data for replication.

Configuring a Peer Relationship

> **Note:** If your cluster uses SAML Authentication, see Configuring Peers with SAML Authentication on page 479 before configuring a peer.

1. Go to the **Peers** page by selecting **Backup** > **Peers**. If there are no existing peers, you will see only an **Add Peer** button in addition to a short message. If peers already exist, they display in the Peers list.
2. Click the **Add Peer** button.
3. In the **Add Peer** dialog box, provide a name, the URL (including the port) of the Cloudera Manager Server 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 TLS/SSL be used. A warning is shown if the URL scheme is `http` instead of `https`. After configuring both peers to use TLS/SSL, add the remote source Cloudera Manager TLS/SSL certificate to the local Cloudera Manager truststore, and vice versa. See Configuring TLS (Encryption Only) for Cloudera Manager.
4. Click the **Add Peer** button in the dialog box to create the peer relationship.

   The peer is added to the Peers list. Cloudera Manager automatically tests the connection between the Cloudera Manager Server and the peer. You can also click **Test Connectivity** to test the connection.

Modifying Peers

1. Go to the **Peers** page by selecting **Backup** > **Peers**. If there are no existing peers, you will see only an **Add Peer** button in addition to a short message. If peers already exist, they display in the Peers list.
2. Do one of the following:
   - **Edit**
     1. In the row for the peer, select **Edit**.
     2. Make your changes.
     3. Click **Update Peer** to save your changes.
   - **Delete** - In the row for the peer, click **Delete**.

Configuring an External Account for Amazon S3 Replication

To configure Amazon S3 as a source or destination for HDFS replication, configure an **External Account** that specifies the type of authentication to use. Amazon offers two types of authentication you can use with Amazon S3:

- IAM Authentication. This type of authentication presumes that clients can have access to all of the data.

- Access Key / Secret Key. This type of authentication is better suited for environments where you have multiple users or multi-tenancy. You can configure selective access for different data paths using this authentication.

For more information, see the Amazon S3 documentation.

To configure an External Account for Amazon S3:

1. Go to **Administration** > **External Accounts**.
2. Click the **Add** button and select one of the following:
   - **Access Key Authentication**

     This authentication mechanism requires you to obtain AWS credentials.

     1. Enter a **Name** of your choosing for this account.
     2. Enter the **AWS Access Key ID**.
     3. Enter the **AWS Secret Key**.
   - **IAM Role-Based Authentication**

     Select this option to use Amazon IAM role-base authentication.

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.

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, synchronizing the data set on the *destination* service with the data set on the *source* service, based on a specified replication schedule. You can also replicate HDFS data to and from Amazon S3. The destination service must be managed by the Cloudera Manager Server where the replication is being set up, and the source service can be managed by that same server or by a peer Cloudera Manager Server. You can also replicate HDFS data within a cluster by specifying different source and destination directories.

Remote BDR Replication automatically copies HDFS metadata to the destination cluster as it copies files. HDFS metadata need only be backed up locally. For information about how to backup HDFS metadata locally, see Backing Up and Restoring NameNode Metadata on page 174.

> **Important:** To use HDFS replication, both the destination 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 502.

Source Data

While a replication runs, ensure that the source directory is not modified. A file added during replication does not get replicated. If you delete a file during replication, the replication fails.

Additionally, ensure that all files in the directory are closed. Replication fails if source files are open. If you cannot ensure that all source files are closed, you can configure the replication to continue despite errors. Uncheck the **Abort on Error** option for the HDFS replication. For more information, see Configuring Replication of HDFS Data on page 480

After the replication completes, you can view the log for the replication to identify opened files. Ensure these files are closed before the next replication runs.

Network Latency and Replication

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

Performance and Scalability Limitations

HDFS replication has the following limitations:

- Maximum number of files for a single replication job: 100 million.
- Maximum number of files for a replication schedule that runs more frequently than once in 8 hours: 10 million.
- The throughput of the replication job depends on the absolute read and write throughput of the source and destination clusters.
- Regular rebalancing of your HDFS clusters is required for efficient operation of replications. See HDFS Balancers on page 183.

Note: Cloudera Manager provides downloadable data that you can use to diagnose HDFS replication performance. See Monitoring the Performance of HDFS Replications on page 490.

Configuring Replication of HDFS Data

1. Verify that your cluster conforms to one of the Supported Replication Scenarios.
2. If you are using different Kerberos principals for the source and destination clusters, add the *destination* principal as a proxy user on the *source* cluster. For example, if you are using the `hdfssrc` principal on the source cluster and the `hdfsdest` principal on the destination cluster, add the following properties to the HDFS service **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 the client configuration and restart all services on the *source* cluster.

3. If the source cluster is managed by a different Cloudera Manager server than the destination cluster, configure a peer relationship. If the source or destination is Amazon S3, you must configure an External Account.
4. Do one of the following:
   1. Select **Backup** > **Replication Schedules**
   2. Click **Create Schedule** > **HDFS Replication**.

   or

   1. Select **Clusters** > ***HDFS Service Name***.
   2. Select **Quick Links** > **Replication**.
   3. Click **Create Schedule** > **HDFS Replication**.

   The **Create HDFS Replication** dialog box displays, and opens displaying the **General** tab. Click the **Peer** or **External Accounts** link if your replication job requires them and you need to create these entities.

5. Select the **General** tab to configure the following:

a. Click the **Source** field and select the source HDFS service. You can select HDFS services managed by a peer Cloudera Manager Server, local HDFS services (managed by the Cloudera Manager Server for the Admin Console you are logged into), or an External Account, such as Amazon S3.
b. Enter the **Source Path** to the directory (or file) you want to replicate.For replication to Amazon S3, enter the path using the following form:

```
s3a://bucket name/path
```

c. Click the **Destination** field and select the destination HDFS service from the HDFS services managed by the Cloudera Manager Server for the Admin Console you are logged into, or select an External Account.
d. Enter the **Destination Path** where the source files should be saved. For replication to Amazon S3, enter the path using the following form:

```
s3a://bucket name/path
```

e. Select a **Schedule**:

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

f. Enter the user to run the replication job in the **Run As Username** field. By default this is `hdfs`. If you want to run the job as a different user, enter the user name here. If you are using Kerberos, you *must* provide a user name here, and it must be one with an ID greater than 1000. (You can also configure the minimum user ID number with the **min.user.id** property in the YARN or MapReduce service.) Verify that the user running the job has a home directory, `/user/`*`username`*, owned by *`username`*`:supergroup` in HDFS. This user must have permissions to read from the source directory and write to the destination directory.

Note the following:

- The User must not be present in the list of banned users specified with the **Banned System Users** property in the YARN configuration (Go to the YARN service, select **Configuration** tab and search for the property). For security purposes, the `hdfs` user is banned by default from running YARN containers.
- The requirement for a user ID that is greater than 1000 can be overridden by adding the user to the "white list" of users that is specified with the **Allowed System Users** property. (Go to the YARN service, select **Configuration** tab and search for the property.)

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

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

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

- **Maximum Map Slots** - Limits for the number of map slots per mapper. The default value is 20.
- **Maximum Bandwidth** - Limits for the bandwidth per mapper. The default is 100 MB.
- **Replication Strategy** - Whether file replication tasks should be distributed among the mappers statically or dynamically. (The default is **Dynamic**.) Static replication distributes file replication tasks among the mappers up front to achieve a uniform distribution based on the file sizes. Dynamic replication distributes file replication tasks in small sets to the mappers, and as each mapper completes its tasks, it dynamically acquires and processes the next unallocated set of tasks. There are additional tuning options you can use to improve performance when using the **Dynamic** strategy. See HDFS Replication Tuning on page 488.

7. Select the **Advanced Options** tab, to configure the following:

- **Add Exclusion** click the link to exclude one or more paths from the replication.

The **Regular Expression-Based Path Exclusion** field displays, where you can enter a regular expression-based path.

Click ✚ to add additional regular expressions.

- **MapReduce Service** - The MapReduce or YARN service to use.
- **Log path** - An alternate path for the logs.
- **Error Handling** You can select the following:
  - **Abort on Error** - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. **Abort on Error** is off by default.
  - **Skip Checksum Checks** - Whether to skip checksum checks on the copied files. If checked, checksums are not validated. Checksums are checked by default.

> **Important:** You must skip checksum checks to prevent replication failure due to nonmatching checksums in the following cases:
>
> - Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster.
> - Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster.
> - Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster.
>
> Checksums are used for two purposes:
>
> - To skip replication of files that have already been copied. If **Skip Checksum Checks** is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination.
> - To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data.

  - **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 destination file system. By default source system settings are preserved. When **Permission** is checked, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When **Extended attributes** is checked, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.)

    If you select one or more of the **Preserve** options and you are replicating *to* Amazon S3, the values all of these items are saved in meta data files on S3. When you replicate *from* Amazon S3 to HDFS, you can select which of these options you want to preserve.

> **Note:** To preserve permissions to HDFS, you must be running as a superuser on the *destination* cluster. Use the "Run As Username" option to ensure that is the case.

See Replication of Encrypted Data on page 504 and HDFS Transparent Encryption.

- **Delete Policy** - Whether files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include:

  - **Keep Deleted Files** - Retains the destination files even when they no longer exist at the source. (This is the default.).
  - **Delete to Trash** - If the HDFS trash is enabled, files are moved to the trash folder. (Not supported when replicating to Amazon S3.)
  - **Delete Permanently** - Uses the least amount of space; use with caution.

- **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.

8. Click **Save Schedule**.

   The replication task now appears as a row in the **Replications Schedule** table. (It can take up to 15 seconds for the task to appear.)

   If you selected **Immediate** in the **Schedule** field, the replication job begins running when you click **Save Schedule**.

To specify additional replication tasks, select **Create** > **HDFS Replication**.

> **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 502.

Limiting Replication to Specific DataNodes

If your cluster has clients installed on hosts with limited resources, HDFS replication may use these hosts to run commands for the replication, which can cause performance degradation. You can limit HDFS replication to run only on selected DataNodes by specifying a "whitelist" of DataNode hosts.

To configure the hosts used for HDFS replication:

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

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

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

Viewing Replication Schedules

The **Replications Schedules** page displays a row of information about each scheduled replication job. Each row also displays recent messages regarding the last time the Replication job ran.

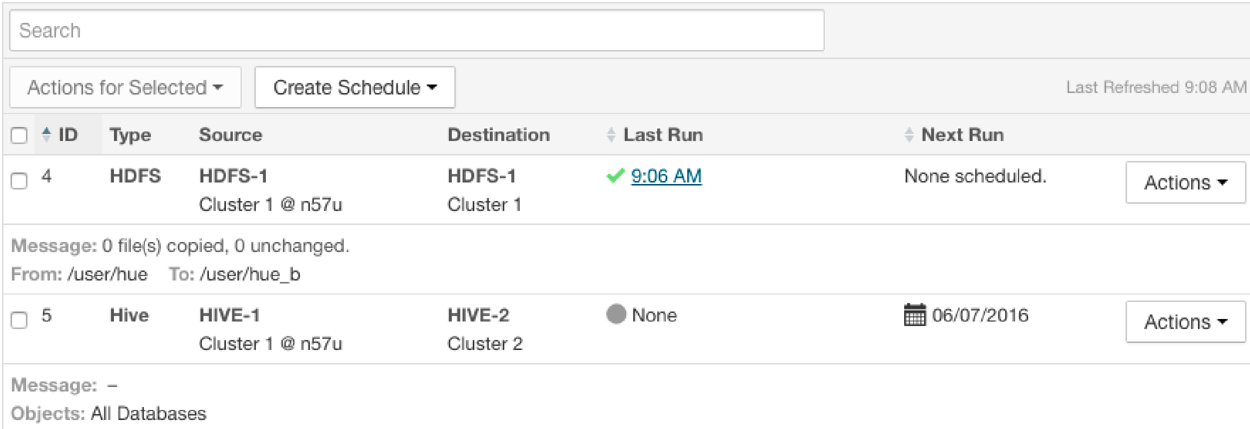


Figure 15: Replication Schedules Table

Only one job corresponding to a replication schedule can occur at a time; if another job associated with that same replication schedule starts before the previous one has finished, the second one is canceled.

You can limit the replication jobs that are displayed by selecting filters on the left. If you do not see an expected schedule, adjust or clear the filters. Use the search box to search the list of schedules for path, database, or table names.

The **Replication Schedules** columns are described in the following table.

Table 27: Replication Schedules Table

Column	Description
ID	An internally generated ID number that identifies the schedule. Provides a convenient way to identify a schedule. Click the **ID** column label to sort the replication schedule table by ID.
Type	The type of replication scheduled, either HDFS or Hive.
Source	The source cluster for the replication.
Destination	The destination cluster for the replication.
Objects	Displays on the bottom line of each row, depending on the type of replication: • **Hive** - A list of tables selected for replication. • **HDFS** - A list of paths selected for replication. For example: 
Last Run	The date and time when the replication last ran. Displays **None** if the scheduled replication has not yet been run. Click the date and time link to view the Replication History page for the replication. Displays one of the following icons: • ✔ - Successful. Displays the date and time of the last run replication.

Column	Description
	• - Failed. Displays the date and time of a failed replication. • - None. This scheduled replication has not yet run. •  - Running. Displays a spinner and bar showing the progress of the replication. Click the **Last Run** column label to sort the Replication Schedules table by the last run date.
Next Run	The date and time when the next replication is scheduled, based on the schedule parameters specified for the schedule. Hover over the date to view additional details about the scheduled replication. Click the **Next Run** column label to sort the Replication Schedules table by the next run date.
Actions	The following items are available from the **Action** button: • **Show History** - Opens the **Replication History** page for a replication. See Viewing Replication History on page 486. • **Edit Configuration** - Opens the **Edit Replication Schedule** page. • **Dry Run** - Simulates a run of the replication task but does not actually copy any files or tables. After a Dry Run, you can select **Show History**, which opens the **Replication History** page where you can view any error messages and the number and size of files or tables that would be copied in an actual replication. • Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for the last 10 runs of the schedule: 1. Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. 2. Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. 3. When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • **Run Now** - Runs the replication task immediately. • **Disable | Enable** - Disables or enables the replication schedule. No further replications are scheduled for disabled replication schedules. • **Delete** - Deletes the schedule. Deleting a replication schedule does not delete copied files or tables.

- While a job is in progress, the **Last Run** column displays a spinner and progress bar, and each stage of the replication task is indicated in the message beneath the job's row. Click the **Command Details** link to view details about the execution of the command.
- If the job is successful, the number of files copied is indicated. If there have been no changes to a file at the source since the previous job, then that file is *not* copied. As a result, after the initial job, only a subset of the files may actually be copied, and this is indicated in the success message.
- If the job fails, the icon displays.
- To view more information about a completed job, select **Actions** > **Show History**. See Viewing Replication History on page 486.

Enabling, Disabling, or Deleting A Replication Schedule

When you create a new replication schedule, it is automatically enabled. If you disable a replication schedule, it can be re-enabled at a later time.

To enable, disable, or delete a replication schedule, do one of the following:

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

-or-

- 1. Select one or more replication schedules in the table by clicking the check box the in the left column of the table.
  2. Click **Actions for Selected** > **Enable|Disable|Delete**.

Viewing Replication History

You can view historical details about replication jobs on the **Replication History** page.

To view the history of a replication job:

1. Select **Backup** > **Replication Schedules** to go to the **Replication Schedules** page.
2. Locate the row for the job.
3. Click **Actions** > **Show History**.

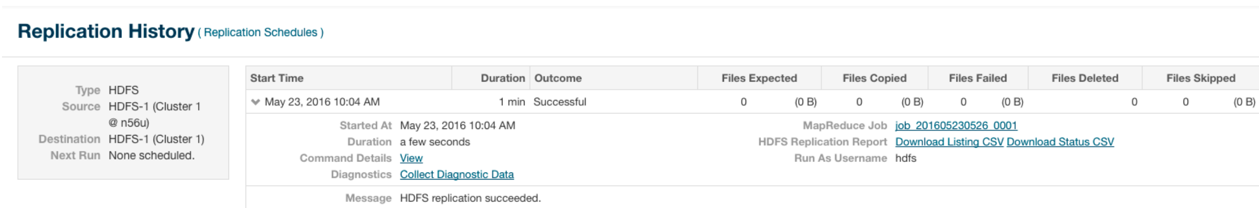


Figure 16: Replication History Screen (HDFS)

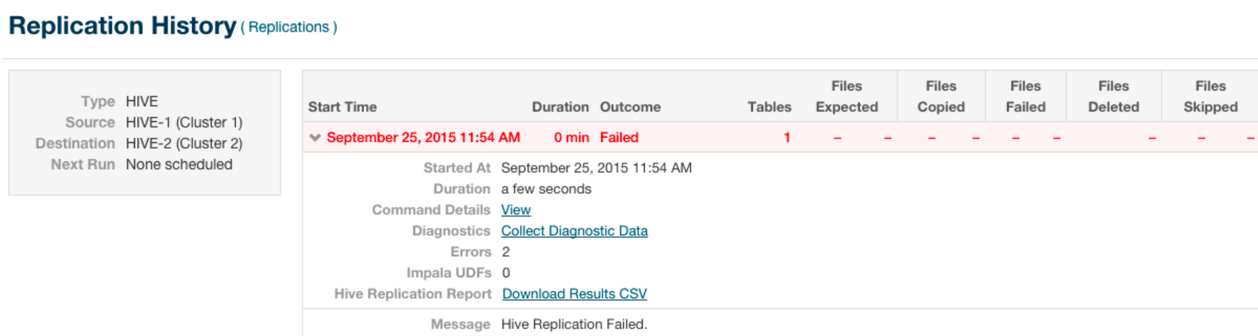


Figure 17: Replication History Screen (Hive, Failed Replication)

The **Replication History** page displays a table of previously run replication jobs with the following columns:

Table 28: Replication History Table

Column	Description
Start Time	Time when the replication job started. Click › to expand the display and show details of the replication. In this screen, you can: • Click the **View** link to open the **Command Details** page, which displays details and messages about each step in the execution of the command. Click › to expand the display for a **Step** to: – View the actual command string. – View the Start time and duration of the command. – Click the **Context** link to view the service status page relevant to the command. – Select one of the tabs to view the **Role Log**, **stdout**, and **stderr** for the command. See Viewing Running and Recent Commands.

Column	Description
	• Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for this run of the schedule: 1. Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. 2. Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. 3. When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • (HDFS only) Link to view details on the **MapReduce Job** used for the replication. See Viewing and Filtering MapReduce Activities. • (Dry Run only) View the number of **Replicable Files**. Displays the number of files that would be replicated during an actual replication. • (Dry Run only) View the number of **Replicable Bytes**. Displays the number of bytes that would be replicated during an actual replication. • Link to download a CSV file containing a **Replication Report**. This file lists the databases and tables that were replicated. • View the number of **Errors** that occurred during the replication. • View the number of **Impala UDFs** replicated. (Displays only for Hive replications where **Replicate Impala Metadata** is selected.) • Click the link to download a CSV file containing a **Download Listing**. This file lists the files and directories that were replicated. • Click the link to download a CSV file containing **Download Status**. • If a user was specified in the **Run As Username** field when creating the replication job, the selected user displays. • View messages returned from the replication job.
Duration	Amount of time the replication job took to complete.
Outcome	Indicates success or failure of the replication job.
Files Expected	Number of files expected to be copied, based on the parameters of the replication schedule.
Files Copied	Number of files actually copied during the replication.
Tables	(Hive only) Number of tables replicated.
Files Failed	Number of files that failed to be copied during the replication.
Files Deleted	Number of files that were deleted during the replication.
Files Skipped	Number of files skipped during the replication. The replication process skips files that already exist in the destination and have not changed.

Backing Up NameNode Metadata

This section describes how to back up and restore NameNode metadata.

1. Make a single backup of the VERSION file. This does not need to be backed up regularly as it does not change, but it is important since it contains the clusterID along with other details.
2. Do not use the http://<namenode>:50070/getimage?getimage=1&txid=latest directly. This is considered an internal API call and is subject to change without notice. It also requires that you know which NameNode is the active one. Instead, use the following command and it will automatically determine the active NN and retrieve the current fsimage and place it in the backup_dir defined.

```
$ hdfs dfsadmin -fetchImage backup_dir
```

3. If both
    a. Add the new host to the cluster and add the NameNode role to the host. Make sure it has the same hostname as original NN. NameNode and SNN were to suddenly die and a new one needs to be created, the general restore process is listed below.
    b. Create the appropriate directory path for the NameNode name.dir (e.g. /dfs/nn/current), ensuring that the permissions are set correctly.
    c. Copy the VERSION and latest fsimage file to the "current" directory
    d. Run `md5sum fsimage > fsimage.md5` to create the md5 file for the fsimage. This could have also been done when the fsimage file was originally backed up if you prefer.
    e. Start the NameNode process.

Upon startup, the NameNode process will read the fsimage file and commit it to memory. If the journal nodes are up and running still and there are edits files present, any edits newer than the fsimage will also be applied. If the journal nodes are also unavailable, then this is the scenario where you could lose up to 1 hour's worth of data.

HDFS Replication to Amazon S3

You can use Cloudera Manager to replicate HDFS data to and from Amazon S3, however you cannot replicate data from one Amazon S3 instance to another using Cloudera Manager. You must have the appropriate credentials to access the Amazon S3 account and you must create buckets in Amazon S3 to store the replicated files.

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

To configure HDFS replication to Amazon S3:

1. Create an **External Account** using Amazon S3 credentials. See Configuring an External Account for Amazon S3 Replication on page 478.
2. Create an **HDFS Replication Schedule**. See HDFS Replication on page 479.

HDFS Replication Tuning

When you create a HDFS replication job, Cloudera recommends setting the **Replication Strategy** to **Dynamic** to improve performance. The dynamic strategy operates by creating chunks of references to files and directories under the **Source Path** defined for the replication, and then allowing the mappers to continually request chunks one-by-one from a logical queue of these chunks. In addition, chunks can contain references to files whose replication can be skipped because they are already up-to-date on the destination cluster. The other option for setting the **Replication Strategy** is **Static.** Static replication distributes file replication tasks among the mappers up front to achieve a uniform distribution based on the file sizes and may not provide optimal performance.

When you use the **Dynamic** strategy, you can improve the performance of HDFS replication by configuring the way chunks are created and packed with file references, which is particularly important for replication jobs where the number of files is very high (1 million or more), relative to the number of mappers.

You can improve the performance of HDFS replication for these types of replications by choosing one of the following options:

Files per Chunk

When you have 1 million files or more in your source cluster that are to be replicated, Cloudera recommends increasing the chunking defaults so that you have no more than 50 files per chunk. This ensures you do not have significant "long-tail behavior," such as when a small number of mappers executing the replication job are working on large copy tasks, but other mappers have already finished. This is a global configuration that applies to all replications, and is disabled by default.

To configure the number of files per Chunk:

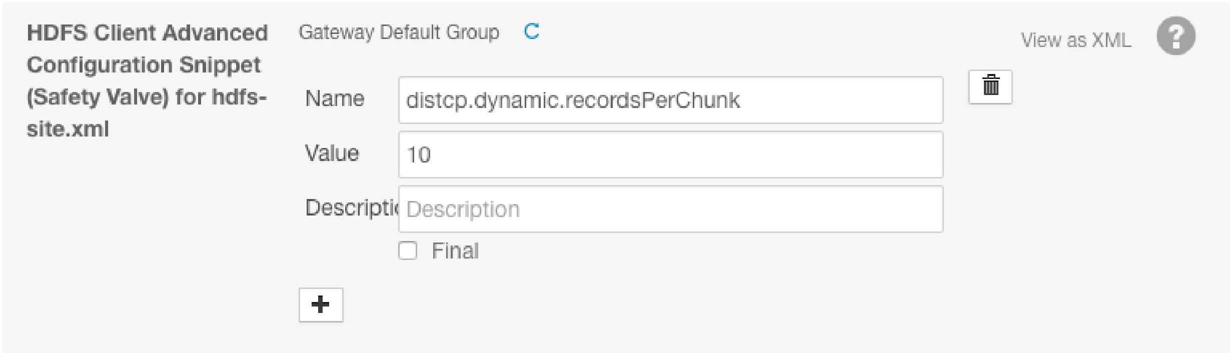
1. Open the Cloudera Manager Admin Console for the destination cluster and go to ***Cluster Name*** > **Configuration**.
2. Search for the **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** property.
3. Click ✚ to add a new configuration.

4. Add the following property:

   `distcp.dynamic.recordsPerChunk`

   Cloudera recommends that you start with a value of 10. Set this number so that you have no more than 50 files per chunk.

   For example:



Chunk by Size

This option overrides the **Files by Chunk** option. Any value set for `distcp.dynamic.recordsPerChunk` is ignored.

The effective amount of work each mapper performs is directly related to the total size of the files it has to copy. You can configure HDFS replications to distribute files into chunks for replication based on the chunk size. This is a global configuration that applies to all replications, and is disabled by default. This option helps improve performance when the sizes of the files you are replicating vary greatly and you expect that most of the files in those chunks have been modified and will need to be copied during the replication.

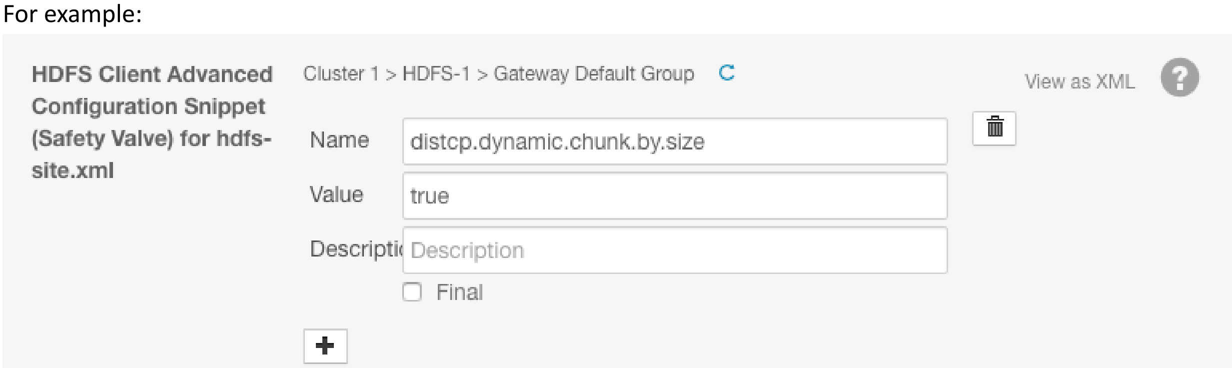
This **Chunk by Size** option can induce "long-tail behavior" (where a small number of mappers executing the replication job are working on large copy tasks, but other mappers have already finished) if a significant percentage of your larger files rarely change. This could, for example, cause some chunks to have only a single file that does not get copied because it did not change, which causes wasted execution time for the mappers.

To enable Chunk by Size:

1. Open the Cloudera Manager Admin Console for the destination cluster and go to ***Cluster Name*** > **Configuration**.
2. Search for the **HDFS Client Advanced Configuration Snippet (Safety Valve) for hdfs-site.xml** property.
3. Click ✚ to add a new configuration.
4. Add the following property and set its value to `true`:

   `distcp.dynamic.chunk.by.size`

   For example:

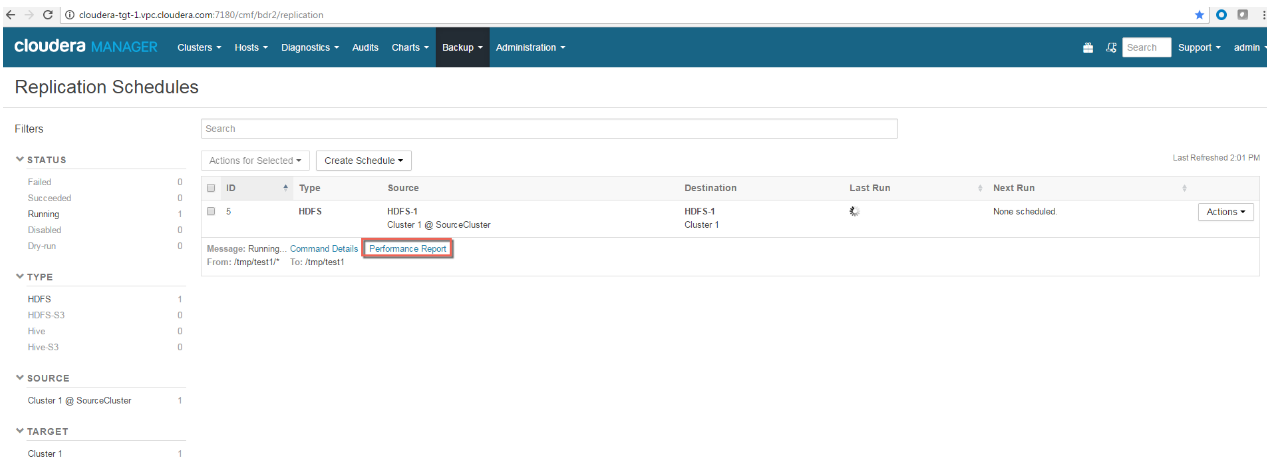


Monitoring the Performance of HDFS Replications

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

To view the performance data for a *running* HDFS replication schedule:

1. Go to **Backup** > **Replication Schedules**.
2. Locate the row for the schedule.
3. Click the **Performance Report** link to download the CSV file.
4. To view the data, import the file into a spreadsheet program such as Microsoft Excel.

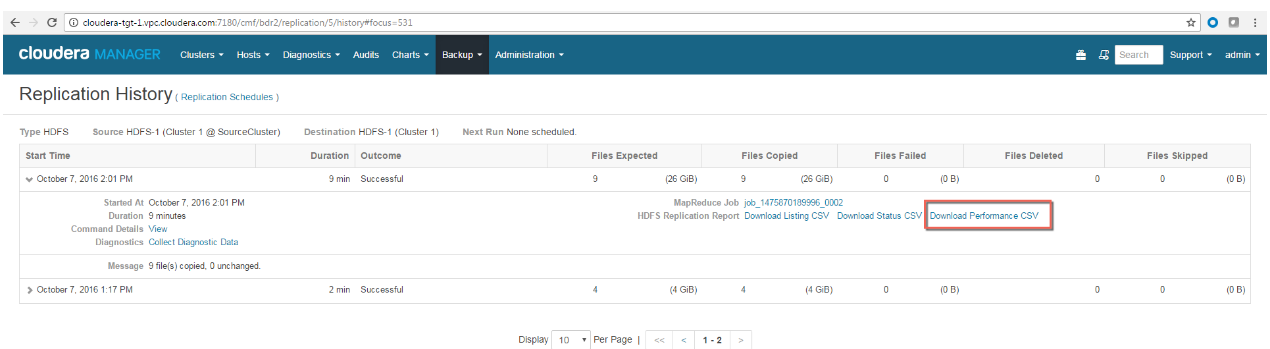


To view the performance data for a *completed* HDFS replication schedule:

1. Go to **Backup** > **Replication Schedules**.
2. In the row for the schedule, click **Actions** > **Show History**.

   The **Replication History** page for the replication schedule displays.

3. Click › to expand the display.
4. Click the **Download Performance CSV** link to download the CSV file.
5. To view the data, import the file into a spreadsheet program such as Microsoft Excel.



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

The information provided by the CSV link is a summary that contains the last report of a given MapReduce job. The data returned by the CSV files downloaded from the Cloudera Manager Admin console has the following structure:

Table 29:

Performance Data Columns	Description
Timestamp	Time when the performance data was collected
Host	Name of the host where the MapReduce job was running.
SrcFile	Name of the source file being copied by the MapReduce job.
TgtFile	Name of the file to which the source file was being copied on the target.
BytesCopied	Number of bytes copied so far.
Time	Total time elapsed for this copy operation.
CurrThroughput	Current throughput in bytes per second.
AvgThroughput	Average throughput in bytes per second since the start of this file transfer.
TotalSleepTime	Number of seconds the transfer was stalled due to throughput throttling. This is expected to be zero unless the throughput was throttled using the **Maximum Bandwidth** parameter for the replication schedule. (You configure his parameter on the **Advanced** tab when creating or editing a replication schedule.)

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

	A	B	C	D	E	F	G	H	I
1	Timestamp	Host	SrcFile	TgtFile	BytesCopied	Time	CurrThroughput	AvgThroughput	TotalSleepTime
2	07:02.3	cloudera-tgt-3.vpc.cloudera.com	hdfs://cloudera-src-1.vpc.cloudera.com:8020/tmp/test1/small.3	hdfs://cloudera-tgt-1.vpc.cloudera.com:8020/tmp/test1/.distcp.tmp.8	8192	27[ms]	315076	303407	0
3	07:39.5	cloudera-tgt-4.vpc.cloudera.com	hdfs://cloudera-src-1.vpc.cloudera.com:8020/tmp/test1/small.4	hdfs://cloudera-tgt-1.vpc.cloudera.com:8020/tmp/test1/.distcp.tmp.2	8192	6[ms]	1365333	1365333	0
4	04:10.2	cloudera-tgt-4.vpc.cloudera.com	hdfs://cloudera-src-1.vpc.cloudera.com:8020/tmp/test1/small.1	hdfs://cloudera-tgt-1.vpc.cloudera.com:8020/tmp/test1/.distcp.tmp.7	8192	5[ms]	1638400	1638400	0
5									

Note the following limitations and known issues:

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

	A	B	C	D	E	F	G	H	I	J	K	L
1	Timestamp	Host	SrcFile	TgtFile	BytesCopied	Time	CurrThroughput	AvgThroughput	TotalSleepTime			
2	No performance statistics available yet: please try again later.											
3												
4												
5												
6												
7												

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

Hive Replication

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

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

Configuration notes:

- If the `hadoop.proxyuser.hive.groups` configuration has been changed to restrict access to the Hive Metastore Server to certain users or groups, the `hdfs` group or a group containing the `hdfs` user must also be included in the list of groups specified for Hive replication to work. This configuration can be specified either on the Hive service as an override, or in the core-site HDFS configuration. This applies to configuration settings on both the source and destination clusters.

- If you configured Synchronizing HDFS ACLs and Sentry Permissions on the target cluster for the directory where HDFS data is copied during Hive replication, the permissions that were copied during replication, are overwritten by the HDFS ACL synchronization and are not preserved
- If you are using Kerberos to secure your clusters, see Enabling Replication Between Clusters in Different Kerberos Realms on page 502 for details about configuring it.

Network Latency and Replication

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

Host Selection for Hive Replication

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

To configure the hosts used for Hive Replication:

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

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

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

Hive Tables and DDL Commands

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

- If you configure replication of a Hive table and then later drop that table, the table remains on the destination cluster. The table is not dropped when subsequent replications occur.
- If you drop a table on the destination cluster, and the table is still included in the replication job, the table is re-created on the destination during the replication.
- If you drop a table partition or index on the source cluster, the replication job also drops them on the destination cluster.
- If you truncate a table, and the **Delete Policy** for the replication job is set to **Delete to Trash** or **Delete Permanently**, the corresponding data files are deleted on the destination during a replication.

Performance and Scalability Limitations

Hive replication has the following limitations:

- Maximum number of databases: 100
- Maximum number of tables per database: 1,000
- Maximum number of partitions per table: 10,000. See Identify Workload Characteristics That Increase Memory Pressure on page 289.
- Maximum total number of tables (across all databases): 10,000
- Maximum total number of partitions (across all tables): 100,000
- Maximum number of indexes per table: 100

Hive Replication in Dynamic Environments

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

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

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

     Value: `true`
4. Save the changes.
5. Restart the HDFS service.

Configuring Replication of Hive Data

1. Verify that your cluster conforms to one of the Supported Replication Scenarios.
2. If the source cluster is managed by a different Cloudera Manager server than the destination cluster, configure a peer relationship. If the source or destination is Amazon S3, you must configure an External Account.
3. Do one of the following:
   - From the **Backup** tab, select **Replications**.
   - From the **Clusters** tab, go to the Hive service and select **Quick Links** > **Replication**.

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

   > **Note:** If you are replication to or from Amazon S3, follow the steps under Hive Replication To and From Amazon S3 on page 500 before completing these steps.

   a. Use the **Source** drop-down list to select the cluster with the Hive service you want to replicate.
   b. Use the **Destination** drop-down list to select the destination for the replication. If there is only one Hive service managed by Cloudera Manager available as a destination, this is specified as the destination. If more than one Hive service is managed by this Cloudera Manager, select from among them.
   c. Leave **Replicate All** checked to replicate all the Hive databases from the source. To replicate only selected databases, uncheck this option and enter the database name(s) and tables you want to replicate.
      - You can specify multiple databases and tables using the plus symbol to add more rows to the specification.
      - You can specify multiple databases on a single line by separating their names with the pipe (|) character. For example: `mydbname1|mydbname2|mydbname3`.
      - Regular expressions can be used in either database or table fields, as described in the following table:

Regular Expression	Result
`[\w].+`	Any database or table name.
`(?!myname\b).+`	Any database or table except the one named myname.
`db1|db2` `[\w_]+`	All tables of the db1 and db2 databases.

Regular Expression	Result

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

d. Select a **Schedule**:

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

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

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

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

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

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

- **Maximum Map Slots** and **Maximum Bandwidth**- Limits for the number of map slots and for bandwidth per mapper. The default is 100 MB.
- **Replication Strategy** - Whether file replication should be static (the default) or dynamic. Static replication distributes file replication tasks among the mappers up front to achieve a uniform distribution based on file sizes. Dynamic replication distributes file replication tasks in small sets to the mappers, and as each mapper processes its tasks, it dynamically acquires and processes the next unallocated set of tasks.

7. Select the **Advanced** tab to specify an export location, modify the parameters of the MapReduce job that will perform the replication, and set other options. You can select a MapReduce service (if there is more than one in your cluster) and change the following parameters:

- Uncheck the **Replicate HDFS Files** checkbox to skip replicating the associated data files.
- If both the source and destination clusters use CDH 5.7.0 or later, uncheck the **Replicate Impala Metadata** checkbox to avoid redundant replication of Impala metadata. (This option only displays when supported by both source and destination clusters.)

  To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.7 or lower, see Impala Metadata Replication on page 501 for information on when to select this option.

- The **Force Overwrite** option, if checked, forces overwriting data in the destination metastore if incompatible changes are detected. For example, if the destination metastore was modified, and a new partition was added to a table, this option forces deletion of that partition, overwriting the table with the version found on the source.

> **Important:** If the **Force Overwrite** option is not set, and the Hive replication process detects incompatible changes on the source cluster, Hive replication fails. This sometimes occurs with recurring replications, where the metadata associated with an existing database or table on the source cluster changes over time.

- By default, Hive metadata is exported to a default HDFS location (`/user/${user.name}/.cm/hive`) and then imported from this HDFS file to the destination Hive metastore. In this example, `user.name` is the process user of the HDFS service on the *destination* cluster. To override the default HDFS location for this export file, specify a path in the **Export Path** field.

  > **Note:** In a Kerberized cluster, the HDFS principal on the *source* cluster must have `read`, `write`, and `execute` access to the **Export Path** directory on the *destination* cluster.

- By default, Hive HDFS data files (for example, `/user/hive/warehouse/db1/t1`) are replicated to a location relative to "`/`" (in this example, to `/user/hive/warehouse/db1/t1`). To override the default, enter a path in the **HDFS Destination Path** field. For example, if you enter `/ReplicatedData`, the data files would be replicated to `/ReplicatedData/user/hive/warehouse/db1/t1`.
- Select the **MapReduce Service** to use for this replication (if there is more than one in your cluster).
- **Log Path** - An alternative path for the logs.
- **Abort on Error** - Whether to abort the job on an error. By selecting the check box, files copied up to that point remain on the destination, but no additional files will be copied. Abort on Error is off by default.
- **Skip Checksum Checks** - Whether to skip checksum checks, which are performed by default.

  Checksums are used for two purposes:

  - To skip replication of files that have already been copied. If **Skip Checksum Checks** is selected, the replication job skips copying a file if the file lengths and modification times are identical between the source and destination clusters. Otherwise, the job copies the file from the source to the destination.
  - To redundantly verify the integrity of data. However, checksums are not required to guarantee accurate transfers between clusters. HDFS data transfers are protected by checksums during transfer and storage hardware also uses checksums to ensure that data is accurately stored. These two mechanisms work together to validate the integrity of the copied data.

- **Delete Policy** - Whether files that were on the source should also be deleted from the destination directory. Options include:

  – **Keep Deleted Files** - Retains the destination files even when they no longer exist at the source. (This is the default.).
  – **Delete to Trash** - If the HDFS trash is enabled, files are moved to the trash folder. (Not supported when replicating to Amazon S3.)
  – **Delete Permanently** - Uses the least amount of space; use with caution.

- **Preserve** - Whether to preserve the **Block Size**, **Replication Count**, and **Permissions** as they exist on the source file system, or to use the settings as configured on the destination file system. By default, settings are preserved on the source.

  > **Note:** You must be running as a superuser to preserve permissions. Use the "Run As Username" option to ensure that is the case.

- **Alerts** - Whether to generate alerts for various state changes in the replication workflow. You can alert **On Failure**, **On Start**, **On Success**, or **On Abort** (when the replication workflow is aborted).

8. Click **Save Schedule**.

The replication task appears as a row in the **Replications Schedule** table. See Viewing Replication Schedules on page 496.

To specify additional replication tasks, select **Create** > **Hive Replication**.

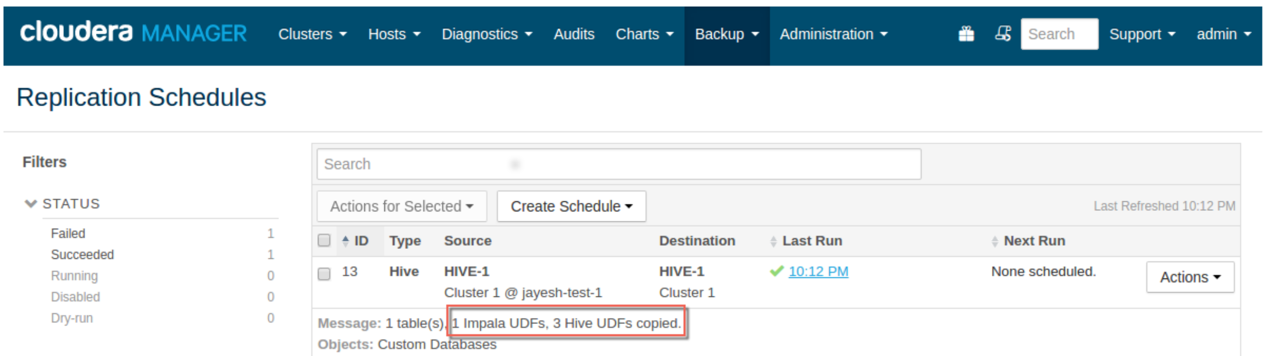
> **Note:** If your replication job takes a long time to complete, and tables change before the replication finishes, the replication may fail. Consider making the **Hive Warehouse Directory** and the directories of any external tables snapshottable, so that the replication job creates snapshots of the directories before copying the files. See Using Snapshots with Replication on page 502.

Replication of Impala and Hive User Defined Functions (UDFs)

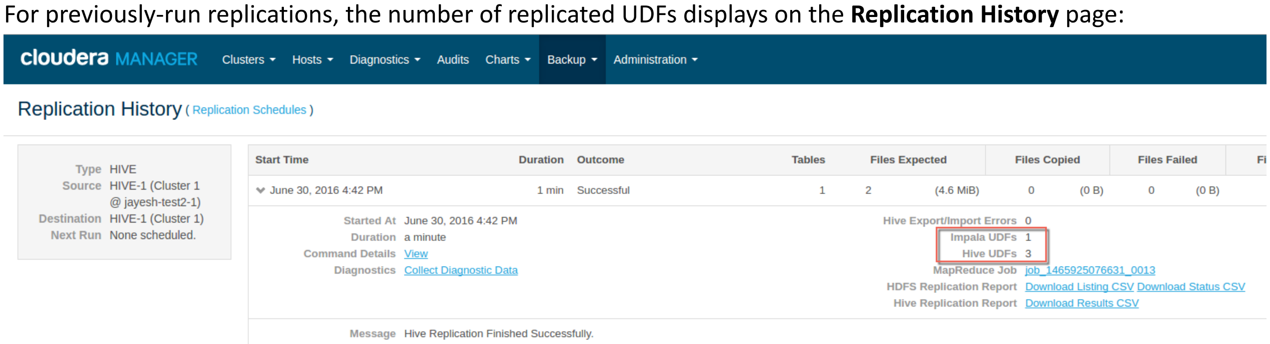
By default, for clusters where the version of CDH is 5.7 or higher, Impala and Hive UDFs are persisted in the Hive Metastore and are replicated automatically as part of Hive replications. See Impala User-Defined Functions (UDFs), Impala Metadata Replication on page 501, and Managing User-Defined Functions (UDFs) with HiveServer2 on page 214.

To replicate Impala UDFs when the version of CDH managed by Cloudera Manager is 5.7 or lower, see Impala Metadata Replication on page 501 for information on when to select the **Replicate Impala Metadata** option on the **Advanced** tab when creating a Hive replication schedule. .

After a replication has run, you can see the number of Impala and Hive UDFs that were replicated during the last run of the schedule on the **Replication Schedules** page:



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



Viewing Replication Schedules

The **Replications Schedules** page displays a row of information about each scheduled replication job. Each row also displays recent messages regarding the last time the Replication job ran.

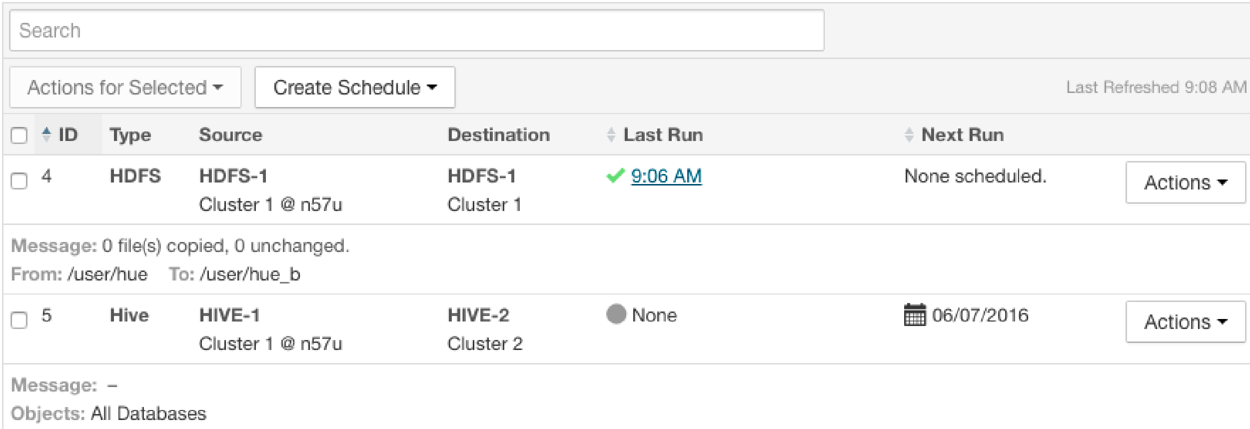


Figure 18: Replication Schedules Table

Only one job corresponding to a replication schedule can occur at a time; if another job associated with that same replication schedule starts before the previous one has finished, the second one is canceled.

You can limit the replication jobs that are displayed by selecting filters on the left. If you do not see an expected schedule, adjust or clear the filters. Use the search box to search the list of schedules for path, database, or table names.

The **Replication Schedules** columns are described in the following table.

Table 30: Replication Schedules Table

Column	Description
ID	An internally generated ID number that identifies the schedule. Provides a convenient way to identify a schedule. Click the **ID** column label to sort the replication schedule table by ID.
Type	The type of replication scheduled, either HDFS or Hive.
Source	The source cluster for the replication.
Destination	The destination cluster for the replication.
Objects	Displays on the bottom line of each row, depending on the type of replication: • **Hive** - A list of tables selected for replication. • **HDFS** - A list of paths selected for replication. For example: 
Last Run	The date and time when the replication last ran. Displays **None** if the scheduled replication has not yet been run. Click the date and time link to view the Replication History page for the replication. Displays one of the following icons: • ✔ - Successful. Displays the date and time of the last run replication.

Column	Description
	• - Failed. Displays the date and time of a failed replication. • - None. This scheduled replication has not yet run. •  - Running. Displays a spinner and bar showing the progress of the replication. Click the **Last Run** column label to sort the Replication Schedules table by the last run date.
Next Run	The date and time when the next replication is scheduled, based on the schedule parameters specified for the schedule. Hover over the date to view additional details about the scheduled replication. Click the **Next Run** column label to sort the Replication Schedules table by the next run date.
Actions	The following items are available from the **Action** button: • **Show History** - Opens the **Replication History** page for a replication. See Viewing Replication History on page 486. • **Edit Configuration** - Opens the **Edit Replication Schedule** page. • **Dry Run** - Simulates a run of the replication task but does not actually copy any files or tables. After a Dry Run, you can select **Show History**, which opens the **Replication History** page where you can view any error messages and the number and size of files or tables that would be copied in an actual replication. • Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for the last 10 runs of the schedule: **1.** Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. **2.** Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. **3.** When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • **Run Now** - Runs the replication task immediately. • **Disable | Enable** - Disables or enables the replication schedule. No further replications are scheduled for disabled replication schedules. • **Delete** - Deletes the schedule. Deleting a replication schedule does not delete copied files or tables.

- While a job is in progress, the **Last Run** column displays a spinner and progress bar, and each stage of the replication task is indicated in the message beneath the job's row. Click the **Command Details** link to view details about the execution of the command.
- If the job is successful, the number of files copied is indicated. If there have been no changes to a file at the source since the previous job, then that file is *not* copied. As a result, after the initial job, only a subset of the files may actually be copied, and this is indicated in the success message.
- If the job fails, the icon displays.
- To view more information about a completed job, select **Actions** > **Show History**. See Viewing Replication History on page 486.

Enabling, Disabling, or Deleting A Replication Schedule

When you create a new replication schedule, it is automatically enabled. If you disable a replication schedule, it can be re-enabled at a later time.

To enable, disable, or delete a replication schedule, do one of the following:

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

-or-

- 1. Select one or more replication schedules in the table by clicking the check box the in the left column of the table.
  2. Click **Actions for Selected** > **Enable|Disable|Delete**.

Viewing Replication History

You can view historical details about replication jobs on the **Replication History** page.

To view the history of a replication job:

1. Select **Backup** > **Replication Schedules** to go to the **Replication Schedules** page.
2. Locate the row for the job.
3. Click **Actions** > **Show History**.

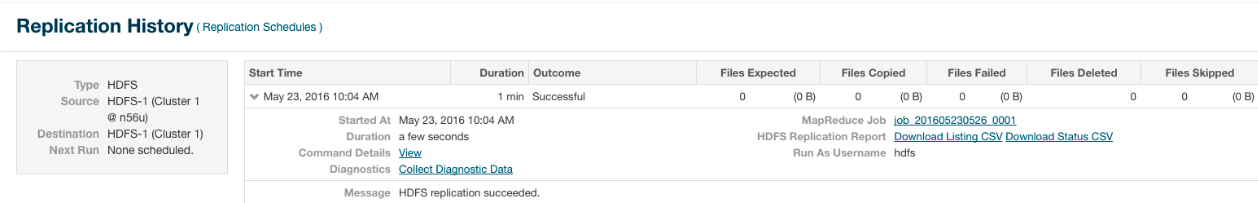


Figure 19: Replication History Screen (HDFS)

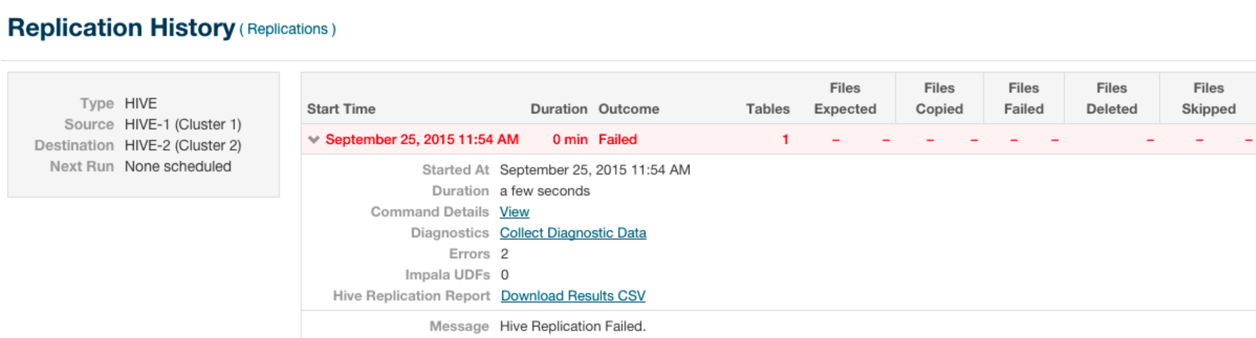


Figure 20: Replication History Screen (Hive, Failed Replication)

The **Replication History** page displays a table of previously run replication jobs with the following columns:

Table 31: Replication History Table

Column	Description
Start Time	Time when the replication job started. Click › to expand the display and show details of the replication. In this screen, you can: • Click the **View** link to open the **Command Details** page, which displays details and messages about each step in the execution of the command. Click › to expand the display for a **Step** to: – View the actual command string. – View the Start time and duration of the command. – Click the **Context** link to view the service status page relevant to the command. – Select one of the tabs to view the **Role Log**, **stdout**, and **stderr** for the command. See Viewing Running and Recent Commands.

Column	Description
	• Click **Collect Diagnostic Data** to open the **Send Diagnostic Data** screen, which allows you to collect replication-specific diagnostic data for this run of the schedule: 1. Select **Send Diagnostic Data to Cloudera** to automatically send the bundle to Cloudera Support. You can also enter a ticket number and comments when sending the bundle. 2. Click **Collect and Send Diagnostic Data** to generate the bundle and open the **Replications Diagnostics Command** screen. 3. When the command finishes, click **Download Result Data** to download a zip file containing the bundle. • (HDFS only) Link to view details on the **MapReduce Job** used for the replication. See Viewing and Filtering MapReduce Activities. • (Dry Run only) View the number of **Replicable Files**. Displays the number of files that would be replicated during an actual replication. • (Dry Run only) View the number of **Replicable Bytes**. Displays the number of bytes that would be replicated during an actual replication. • Link to download a CSV file containing a **Replication Report**. This file lists the databases and tables that were replicated. • View the number of **Errors** that occurred during the replication. • View the number of **Impala UDFs** replicated. (Displays only for Hive replications where **Replicate Impala Metadata** is selected.) • Click the link to download a CSV file containing a **Download Listing**. This file lists the files and directories that were replicated. • Click the link to download a CSV file containing **Download Status**. • If a user was specified in the **Run As Username** field when creating the replication job, the selected user displays. • View messages returned from the replication job.
Duration	Amount of time the replication job took to complete.
Outcome	Indicates success or failure of the replication job.
Files Expected	Number of files expected to be copied, based on the parameters of the replication schedule.
Files Copied	Number of files actually copied during the replication.
Tables	(Hive only) Number of tables replicated.
Files Failed	Number of files that failed to be copied during the replication.
Files Deleted	Number of files that were deleted during the replication.
Files Skipped	Number of files skipped during the replication. The replication process skips files that already exist in the destination and have not changed.

Hive Replication To and From Amazon S3

You can use Cloudera Manager to replicate Hive data and metadata to and from Amazon S3, however you cannot replicate data from one Amazon S3 instance to another using Cloudera Manager. You must have the appropriate credentials to access the Amazon S3 account and you must create buckets in Amazon S3 to store the replicated files.

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

To configure Hive replication to or from Amazon S3:

1. Create an **External Account** using Amazon S3 credentials. See Configuring an External Account for Amazon S3 Replication on page 478.

2. Select **Backup** > **Replication Schedules**.
3. Click **Create Schedule** > **Hive Replication**
4. To back up data to S3:

   a. Select the Source cluster from the **Source** drop-down list.
   b. Select the Amazon S3 destination (one of the External Accounts you created for Amazon S3) from the **Destination** drop-down list.
   c. Enter the path where the data should be copied to in S3. Enter using the following form:

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

   d. Select one of the following **Replication Options**:

      - **Metadata and Data** – Backs up the Hive data from HDFS and its associated metadata.
      - **Metadata only** – Backs up only the Hive metadata.

5. To restore data from S3:

   a. Select the Amazon S3 source (one of the External Accounts you created for Amazon S3) from the **Source** drop-down list.
   b. Select the destination cluster from the **Destination** drop-down list.
   c. Enter the path to the metadata file (`export.json`) where the data should be copied from in S3. Enter using the following form:

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

   d. Select one of the following **Replication Options**:

      - **Metadata and Data** – Restores the Hive data from HDFS from S3 and its associated metadata.
      - **Metadata only** – Restores only the Hive metadata.
      - **Hive-on-S3** – Restores only the Hive tables and references the tables on S3 as a Hive external table. If you drop a table in Hive, the data remains on S3. Only data that was backed up using a Hive/Impala Replication schedule can be restored. However, you can restore a Hive external table that is stored in S3.

6. Complete the configuration of the Hive replication schedule by following the steps under Configuring Replication of Hive Data on page 493, beginning with step 5.c on page 493

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 493. 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, 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 are defined are also replicated.

> **Note:** To run queries or execute DDL statements on tables that have been replicated to a destination cluster, you must run the Impala `INVALIDATE METADATA` statement on the destination cluster to prevent queries from failing. See INVALIDATE METADATA Statement.

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 and Disabling HDFS Snapshots on page 531.

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`.) To access this property:

1. Open Cloudera Manager and browse to the Hive service.
2. Click the **Configuration** tab.
3. In the **Search** box, type `hive.metastore.warehouse.dir`.

   The **Hive Warehouse Directory** property displays.

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 Impala and are replicating any Impala tables using Hive/Impala replication, ensure that the storage locations for the tables and associated databases are also snapshottable. See Enabling and Disabling HDFS Snapshots on page 531.

Enabling Replication Between Clusters in Different Kerberos Realms

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

To enable replication between clusters that reside in different Kerberos realms, additional setup steps are required to ensure that the source and destination clusters can communicate.

> **Note:** If either the source or destination cluster is running Cloudera Manager 4.6 or higher, then both clusters (source and destination) must be running 4.6 or higher. For example, 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 higher.

Ports

When using BDR with Kerberos authentication enabled, BDR requires all the ports listed on the following page: Port Requirements for Backup and Disaster Recovery on page 473.

Additionally, the port used for the Kerberos KDC Server and KRB5 services must be open to all hosts on the destination cluster. By default, this is port 88.

Considerations for Realm Names

If the source and destination clusters each use Kerberos for authentication, use one of the following configurations to prevent conflicts when running replication jobs:

- If the clusters do not use the same KDC (Kerberos Key Distribution Center), Cloudera recommends that you use different realm names for each cluster. Additionally, if you are replicating across clusters in two different realms, see the steps for HDFS and Hive replication later in this topic to setup trust between those clusters.
- You can use the same realm name if the clusters use the same KDC or different KDCs that are part of a unified realm, for example where one KDC is the master and the other is a slave KDC.
- **Note:** If you have multiple clusters that are used to segregate production and non-production environments, this configuration could result in principals that have equal permissions in both environments. Make sure that permissions are set appropriately for each type of environment.

Important: If the source and destination clusters are in the same realm but do not use the same KDC or the KDcs are not part of a unified realm, the replication job will fail.

HDFS Replication

1. On the hosts in the *destination* cluster, ensure that the `krb5.conf` file (typically located at `/etc/kbr5.conf`) on each host has the following information:
    - The kdc information for the *source* cluster's Kerberos realm. For example:

```
[realms]
 SOURCE.MYCO.COM = {
  kdc = src-kdc-1.src.myco.com:88
  admin_server = src-kdc-1.src.myco.com:749
  default_domain = src.myco.com
 }
 DEST.MYCO.COM = {
  kdc = dest-kdc-1.dest.myco.com:88
  admin_server = dest-kdc-1.dest.myco.com:749
  default_domain = dest.myco.com
 }
```

    - Domain/host-to-realm mapping for the *source* cluster NameNode hosts. You configure these mappings in the `[domain_realm]` section. For example, to map two realms named `SRC.MYCO.COM` and `DEST.MYCO.COM`, to the domains of hosts named *hostname*`.src.myco.com` and *hostname*`.dest.myco.com`, make the following mappings in the `krb5.conf` file:

```
[domain_realm]
 .src.myco.com = SRC.MYCO.COM
 src.myco.com = SRC.MYCO.COM
 .dest.myco.com = DEST.MYCO.COM
 dest.myco.com = DEST.MYCO.COM
```

2. On the *destination* cluster, use Cloudera Manager to 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 release is 5.0.1 or lower, 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.

Hive Replication

1. Perform the procedure described in the previous section, 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 *destination* cluster's Kerberos realm.
   - Domain/host-to-realm mapping for the *destination* cluster NameNode hosts.
3. On the *source* cluster, use Cloudera Manager to add the realm of the *destination* 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 destination cluster realm.
   e. Click **Save Changes** to commit the changes.

   It is not necessary to restart any services on the source cluster.

Replication of Encrypted Data

HDFS supports encryption of data at rest (including data accessed through Hive). This topic describes how replication works within and between encryption zones and how to configure replication to avoid failures due to encryption.

Encrypting Data in Transit Between Clusters

A source directory and destination directory may or may not be in an encryption zone. If the destination directory is in an encryption zone, the data on the destination directory is encrypted. If the destination directory is not in an encryption zone, the data on that directory is not encrypted, even if the source directory is in an encryption zone. For more information about HDFS encryption zones, see HDFS Transparent Encryption. Encryption zones are not supported in CDH versions 5.1 or lower.

When you configure encryption zones, you also configure a Key Management Server (KMS) to manage encryption keys. During replication, Cloudera Manager uses TLS/SSL to encrypt the keys when they are transferred from the source cluster to the destination cluster.

When you configure encryption zones, you also configure a Key Management Server (KMS) to manage encryption keys. When a HDFS replication command that specifies an encrypted source directory runs, Cloudera Manager temporarily copies the encryption keys from the source cluster to the destination cluster, using TLS/SSL (if configured for the KMS) to encrypt the keys. Cloudera Manager then uses these keys to decrypt the encrypted files when they are received from the source cluster before writing the files to the destination cluster.

Important: When you configure HDFS replication, you must select the **Skip Checksum check** property to prevent replication failure in the following cases:

- Replications from an encrypted zone on the source cluster to an encrypted zone on a destination cluster.
- Replications from an encryption zone on the source cluster to an unencrypted zone on the destination cluster.
- Replications from an unencrypted zone on the source cluster to an encrypted zone on the destination cluster.

Even when the source and destination 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 destination cluster (using the key for the destination encryption zone). The data transmission is encrypted if you have configured encryption for HDFS Data Transfer.

> **Note:** The decryption and encryption steps happen in the same process on the hosts where the MapReduce jobs that copy the data run. Therefore, data in plain text only exists within the memory of the Mapper task. If a KMS is in use on either the source or destination clusters, and you are using encrypted zones for either the source or destination directories, configure TLS/SSL for the KMS to prevent transferring the key to the mapper task as plain text.

During replication, data travels from the source cluster to the destination cluster using `distcp`. For clusters that use encryption zones, configure encryption of KMS key transfers between the source and destination using TLS/SSL. See Configuring TLS/SSL for the KMS.

To configure encryption of data transmission between source and destination clusters:

- Enable TLS/SSL for HDFS clients on both the source and the destination clusters. For instructions, see Configuring TLS/SSL for HDFS. You may also need to configure trust between the SSL certificates on the source and destination.
- Enable TLS/SSL for the two peer Cloudera Manager Servers. See Configuring TLS (Encryption Only) for Cloudera Manager.
- Cloudera recommends you also enable TLS/SSL communication between the Cloudera Manager Server and Agents. See Configuring TLS Security for Cloudera Manager for instructions.
- Encrypt data transfer using HDFS Data Transfer Encryption. See Configuring Encrypted HDFS Data Transport.

The following blog post provides additional information about encryption with HDFS:
http://blog.cloudera.com/blog/2013/03/how-to-set-up-a-hadoop-cluster-with-network-encryption/.

Security Considerations

The user you specify with the **Run As** field when scheduling a replication job requires full access to both the key and the data directories being replicated. This is not a recommended best practice for KMS management. If you change permissions in the KMS to enable this requirement, you could accidentally provide access for this user to data in other encryption zones using the same key. If a user is not specified in the **Run As** field, the replication runs as the default user, `hdfs`.

To access encrypted data, the user must be authorized on the KMS for the encryption zones they need to interact with. The user you specify with the **Run As** field when scheduling a replication must have this authorization. The key administrator must add ACLs to the KMS for that user to prevent authorization failure.

Key transfer using the KMS protocol from source to the client uses the REST protocol, which requires that you configure TLS/SSL for the KMS. When TLS/SSL is enabled, keys are not transferred over the network as plain text.

See Configuring Encryption.

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. Replication is supported both from CDH 5 to CDH 6 and from CDH 6 to CDH 5, the source and destination cluster do not have to run the same major version of CDH.

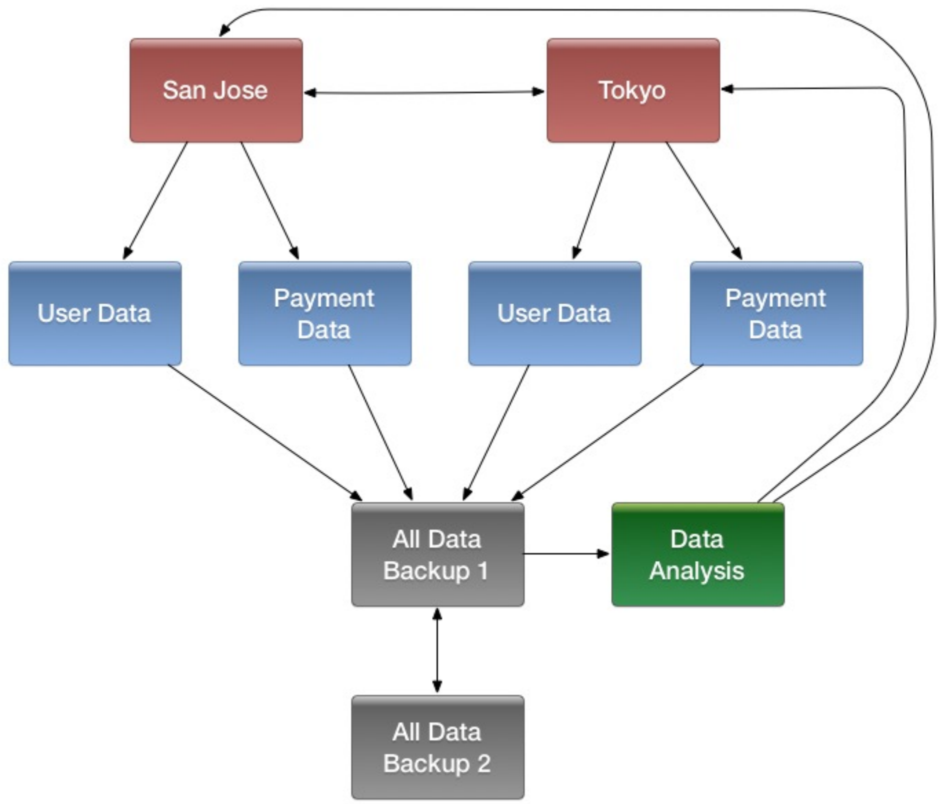
Cluster replication uses an active-push methodology. An HBase cluster can be a source (also called *active*, meaning that it writes new data), a destination (also called *passive*, meaning that it receives data using replication), or can fulfill both roles at once. Replication is asynchronous, and the 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.

Notes 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 write 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. If your tables do not yet exist on the destination cluster, see Creating the Empty Table On the Destination Cluster on page 511.
- 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 comands, 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.
2. 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. See Creating the Empty Table On the Destination Cluster on page 511.
3. On the source cluster, enable replication in Cloudera Manager, or by setting `hbase.replication` to `true` in `hbase-site.xml`.
4. 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
```

5. 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.

6. 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'
```

7. 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
```

8. 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=timestamp1] [--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.

Replicating 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'}
```

Enabling Replication on a Specific Table

To enable replication for a specific table on the source cluster, run the `enable_table_replication <table>` command from the HBase shell on a cluster where a peer has been configured.

Running `enable_table_replication <table>` does the following:

1. Verifies that the table exists on the source cluster.
2. If the table does not exist on the remote cluster, uses the peer configuration to duplicate the table schema (including splits) on the remote cluster.
3. Enables replication on that table.

Configuring Secure Replication

The following steps describe how to set up secure replication between clusters. The steps are the same whether your clusters are all in the same realm or not, with the exception of the last step.

The last step involves setting up custom secure replication configurations per peer. This can be convenient when you need to replicate to a cluster that uses different cross-realm authentication rules than the source cluster. For example, a cluster in Realm A may be allowed to replicate to Realm B and Realm C, but Realm B may not be allowed to replicate to Realm C. If you do not need this feature, skip the last step.

To use this feature, service-level principals and keytabs (specific to HBase) must be specified when you create the cluster peers using HBase Shell.

> **Note:** HBase peer-to-peer replication from a non-Kerberized cluster to a Kerberized cluster is not supported.

1. Set up Kerberos on your cluster, as described in [Enabling Kerberos Authentication Using the Wizard]().
2. If necessary, configure Kerberos cross-realm authentication.
   - At the command line, use the `list_principals` command to list the `kdc`, `admin_server`, and `default_domain` for each realm.
   - Add this information to each cluster using Cloudera Manager. For each cluster, go to **HDFS** > **Configuration** > **Trusted Kerberos Realms**. Add the target and source. This requires a restart of HDFS.
3. Configure ZooKeeper.
4. Configure the following HDFS parameters on both clusters, in Cloudera Manager or in the listed files if you do not use Cloudera Manager:

> **Note:**
>
> If you use Cloudera Manager to manage your cluster, do not set these properties directly in configuration files, because Cloudera Manager will overwrite or ignore these settings. You must set these properties in Cloudera Manager.
>
> For brevity, the Cloudera Manager setting names are not listed here, but you can search by property name. For instance, in the HDFS service configuration screen, search for **dfs.encrypt.data.transfer**. The **Enable Data Transfer Encryption** setting is shown. Selecting the box is equivalent to setting the value to `true`.

```
<!-- In hdfs-site.xml or advanced configuration snippet -->
<property>
  <name>dfs.encrypt.data.transfer</name>
  <value>true</value>
</property>
<property>
  <name>dfs.data.transfer.protection</name>
  <value>privacy</value>
</property>

<!-- In core-site.xml or advanced configuration snippet -->
<property>
  <name>hadoop.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hadoop.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
  <name>hadoop.security.crypto.cipher.suite</name>
  <value>AES/CTR/NoPadding</value>
</property>
<property>
  <name>hadoop.ssl.enabled</name>
  <value>true</value>
</property>
```

5. Configure the following HBase parameters on both clusters, using Cloudera Manager or in `hbase-site.xml` if you do not use Cloudera Managert.

```
<!-- In hbase-site.xml -->
<property>
  <name>hbase.rpc.protection</name>
  <value>privacy</value>
</property>
<property>
```

```
  <name>hbase.thrift.security.qop</name>
  <value>auth-conf</value>
</property>
<property>
  <name>hbase.thrift.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rest.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.ssl.enabled</name>
  <value>true</value>
</property>
<property>
  <name>hbase.security.authentication</name>
  <value>kerberos</value>
</property>
<property>
  <name>hbase.security.authorization</name>
  <value>true</value>
</property>
<property>
  <name>hbase.secure.rpc.engine</name>
  <value>true</value>
</property>
```

6. Add the cluster peers using the simplified `add_peer` syntax, as described in [Add Peer](#).

```
add_peer 'ID', 'CLUSTER_KEY'
```

7. If you need to add any peers which require custom security configuration, modify the `add_peer` syntax, using the following examples as a model.

```
add_peer 'vegas', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
        CONFIG => {'hbase.master.kerberos.principal'          => 'hbase/_HOST@TO_VEGAS',

                 'hbase.regionserver.kerberos.principal'     => 'hbase/_HOST@TO_VEGAS',

                    'hbase.regionserver.keytab.file'            =>
'/keytabs/vegas_hbase.keytab',
                    'hbase.master.keytab.file'                  =>
'/keytabs/vegas_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf1']}

add_peer 'atlanta', CLUSTER_KEY => 'zk1.vegas.example.com:2181:/hbase',
         CONFIG => {'hbase.master.kerberos.principal'           =>
'hbase/_HOST@TO_ATLANTA',
                    'hbase.regionserver.kerberos.principal'     =>
'hbase/_HOST@TO_ATLANTA',
                    'hbase.regionserver.keytab.file'            =>
'/keytabs/atlanta_hbase.keytab',
                    'hbase.master.keytab.file'                  =>
'/keytabs/atlanta_hbase.keytab'},
          TABLE_CFS => {"tbl" => [cf2']}
```

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.

> **Note:** This log accumulation is a powerful side effect of the `disable_peer` command and can be used to your advantage. See [Initiating Replication When Data Already Exists](#) on page 511.

To re-enable the peer, use the command `enable_peer(<"peerID">)`. Replication resumes.

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.

Open the shell on the source cluster and use the `disable_peer` command for each peer, then the `disable_table_replication` command. For example:

```
hbase> disable_peer("1")
hbase> disable_table_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 512.

To start replication again, use the `enable_peer` command.

Creating the Empty Table On the Destination Cluster

If the table to be replicated does not yet exist on the destination cluster, you must create it. The easiest way to do this is to extract the schema using HBase Shell.

1. On the source cluster, `describe` the table using HBase Shell. The output below has been reformatted for readability.

```
hbase> describe acme_users

Table acme_users is ENABLED
acme_users
COLUMN FAMILIES DESCRIPTION
{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => 'FOREVER', KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

2. Copy the output and make the following changes:

   - For the TTL, change `FOREVER` to `org.apache.hadoop.hbase.HConstants::FOREVER`.
   - Add the word `CREATE` before the table name.
   - Remove the line `COLUMN FAMILIES DESCRIPTION` and everything above the table name.

   The result will be a command like the following:

```
CREATE cme_users

{NAME => 'user', DATA_BLOCK_ENCODING => 'NONE', BLOOMFILTER => 'NONE',
REPLICATION_SCOPE => '0', VERSIONS => '3', COMPRESSION => 'NONE',
MIN_VERSIONS => '0', TTL => 'org.apache.hadoop.hbase.HConstants::FOREVER',
KEEP_DELETED_CELLS => 'FALSE',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'false'}
```

3. On the destination cluster, paste the command from the previous step into HBase Shell to create the table.

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.

One way to do this is to take advantage of the write accumulation that happens when a replication peer is disabled.

1. Start replication.
2. Add the destination cluster as a peer and immediately disable it using `disable_peer`.
3. On the source cluster, take a snapshot of the table and export it. The snapshot command flushes the table from memory for you.
4. On the destination cluster, import and restore the snapshot.
5. Run `enable_peer` to re-enable the destination cluster.

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. The following is one way to accomplish this:

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 511.
4. Add the sink as a replication peer to the source, using the `add_peer` command as discussed in Deploying HBase Replication on page 507. 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.

Verifying that Replication is Working

To verify that HBase replication is working, follow these steps to confirm data has been replicated from a source cluster to a remote destination cluster.

1. Install and configure YARN on the source cluster.

   If YARN cannot be used in the source cluster, configure YARN on the destination cluster to verify replication. If neither the source nor the destination clusters can have YARN installed, you can configure the tool to use local mode; however, performance and consistency could be negatively impacted.

2. Make sure that you have the required permissions:

   - You have sudo permissions to run commands as the hbase user, or a user with admin permissions on both clusters.
   - You are an hbase user configured for submitting jobs with YARN.

   > **Note:** To use the hbase user in a secure cluster, use Cloudera Manager to add the hbase user as a YARN whitelisted user. If you are running Cloudera Manager 5.8 or higher, and are running a new installation, the hbase user is already added to the whitelisted users. In addition, /user/hbase should exist on HDFS and owned as the hbase user, because YARN will create a job staging directory there.

3. Run the `VerifyReplication` command:

```
src-node$ sudo -u hbase hbase
org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication peer1 table1
...
       org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication$Verifier$Counters

                BADROWS=2
                CONTENT_DIFFERENT_ROWS=1
                GOODROWS=1
                ONLY_IN_PEER_TABLE_ROWS=1
        File Input Format Counters
                Bytes Read=0
        File Output Format Counters
                Bytes Written=0
```

The following table describes the `VerifyReplication` counters:

Table 32: VerifyReplication Counters

Counter	Description
GOODROWS	Number of rows. On both clusters, and all values are the same.
CONTENT_DIFFERENT_ROWS	The key is the same on both source and destination clusters for a row, but the value differs.
ONLY_IN_SOURCE_TABLE_ROWS	Rows that are only present in the source cluster but not in the destination cluster.
ONLY_IN_PEER_TABLE_ROWS	Rows that are only present in the destination cluster but not in the source cluster.
BADROWS	Total number of rows that differ from the source and destination clusters; the sum of CONTENT_DIFFERENT_ROWS + ONLY_IN_SOURCE_TABLE_ROWS + ONLY_IN_PEER_TABLE_ROWS

By default, `VerifyReplication` compares the entire content of `table1` on the source cluster against `table1` on the destination cluster that is configured to use the replication peer `peer1`.

Use the following options to define the period of time, versions, or column families

Table 33: VerifyReplication Counters

Option	Description
--starttime=<timestamp>	Beginning of the time range, in milliseconds. Time range is forever if no end time is defined.
--endtime=<timestamp>	End of the time range, in milliseconds.
--versions=<versions>	Number of cell versions to verify.
--families=<cf1,cf2,..>	Families to copy; separated by commas.

The following example, verifies replication only for rows with a timestamp range of one day:

```
src-node$ sudo -u hbase hbase
org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication --starttime=1472499077000
 --endtime=1472585477000 --families=c1  peer1 table1
```

Replication Caveats

- Two variables govern replication: `hbase.replication` as described above under Deploying HBase Replication on page 507, and a replication znode. Stopping replication (using `disable_table_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.

Snapshots

You can create HBase and HDFS snapshots using 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 higher.
- HDFS snapshots allow you to create point-in-time backups of directories or the entire filesystem without actually cloning the data. They can improve data replication performance and prevent errors caused by changes to a source directory. These snapshots appear on the filesystem as read-only directories that can be accessed just like other ordinary directories. HDFS snapshots are supported for clusters running CDH 5 or higher. CDH 4 does not support snapshots for HDFS.

NEW! View a video about Using Snapshots and Cloudera Manager to Back Up Data.

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 seven daily snapshots and five weekly snapshots should be maintained.

> **Note:** You can improve the reliability of Data Replication on page 474 by also using snapshots. See Using Snapshots with Replication on page 502.

Managing Snapshot Policies

> **Note:** You must enable an HDFS directory for snapshots to allow snapshot policies to be created for that directory. To designate a HDFS directory as snapshottable, follow the procedure in Enabling and Disabling HDFS Snapshots on page 531.

To create a snapshot policy:

1. Select **Backup** > **Snapshots** in the top navigation bar.

   Existing snapshot policies are shown in a table. See Snapshot Policies Page on page 517.

2. To create a new policy, click **Create snapshot policy**.
3. From the drop-down list, select the service (HDFS or HBase) and cluster for which you want to create a policy.
4. Provide a name for the policy and, optionally, a description.
5. Specify the directories or tables to include in the snapshot.

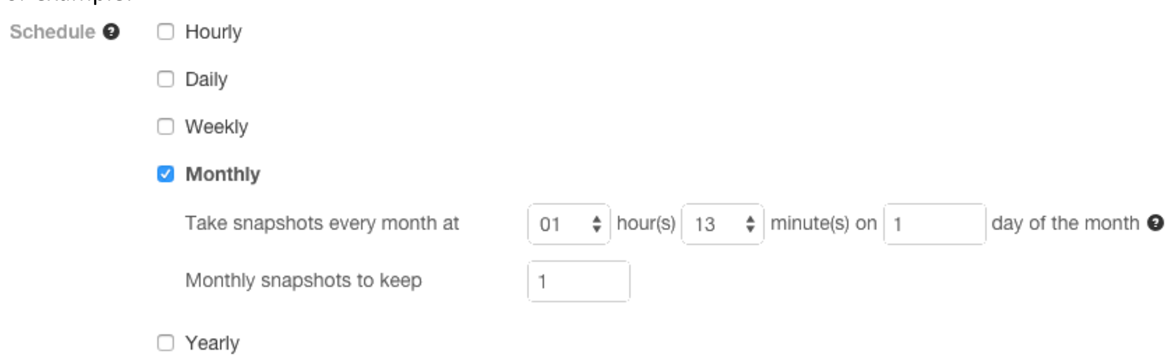
   - For an HDFS service, select the paths of the directories to include in the snapshot. The drop-down list allows you to select only directories that are enabled for snapshotting. If no directories are enabled for snapshotting, a warning displays.

     Click + to add a path and − to remove a path.

   - For an HBase service, list the tables to include in your snapshot. You can use a Java regular expression to specify a set of tables. For example, `finance.*` matchs all tables with names starting with `finance`. You can also create a snapshot for all tables in a given namespace, using the `{`*`namespace`*`}:.*` syntax.

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 select, 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 20:30.

   To select an interval, check its box. Fields display where you can edit the time and number of snapshots to keep. For example:



7. 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.
8. Click **Save Policy**.

   The new Policy displays on the **Snapshot Policies** page. See Snapshot Policies Page on page 517.

To edit or delete a snapshot policy:

1. Select **Backup** > **Snapshots** in the top navigation bar.

   Existing snapshot policies are shown in a table. See Snapshot Policies Page on page 517.
2. Click the **Actions** menu shown next to a policy and select **Edit** or **Delete**.

Snapshot Policies Page

The policies you add are shown in a table on the **Snapshot Policies** screen. The table displays the following columns:

Table 34: Snapshot Policies

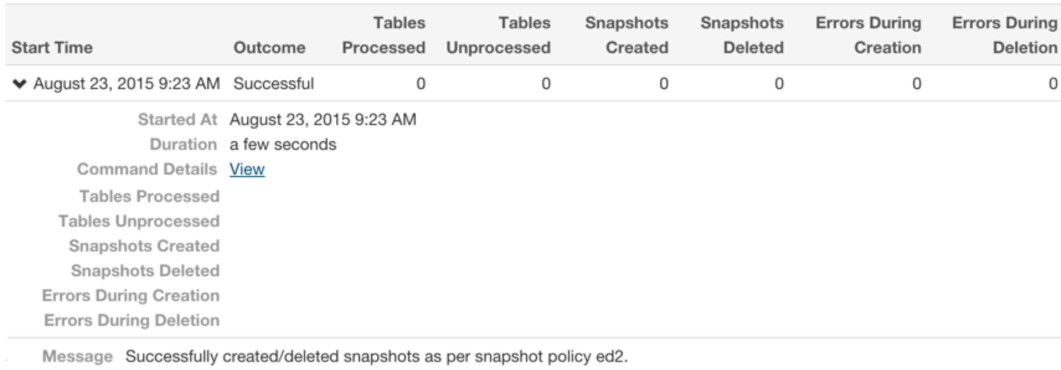
Column	Description
Policy Name	The name of the policy.
Cluster	The cluster that hosts the service (HDFS or HBase).
Service	The service from which the snapshot is taken.
Objects	HDFS Snapshots: The directories included in the snapshot. HBase Snapshots: The tables included in the snapshot.
Last Run	The date and time the snapshot last ran. Click the link to view the Snapshots History page. Also displays the status icon for the last run.
Snapshot Schedule	The type of schedule defined for the snapshot: Hourly, Daily, Weekly, Monthly, or Yearly.
Actions	A drop-down menu with the following options: • **Show History** - Opens the **Snapshots History** page. See Snapshots History on page 517. • **Edit Configuration** - Edit the snapshot policy. • **Delete** - Deletes the snapshot policy. • **Enable** - Enables running of scheduled snapshot jobs. • **Disable** - Disables running of scheduled snapshot jobs.

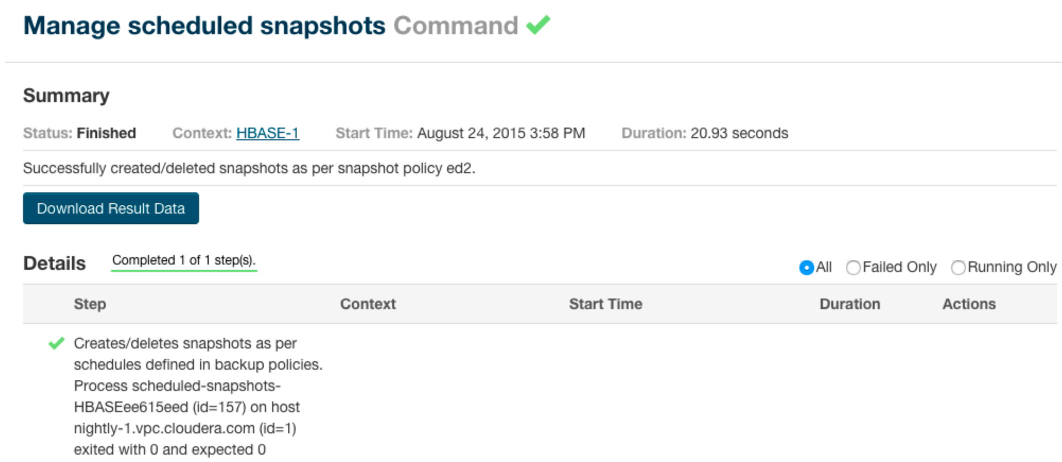
Snapshots History

The **Snapshots History** page displays information about Snapshot jobs that have been run or attempted. The page displays a table of Snapshot jobs with the following columns:

Table 35: Snapshots History

<table>
<tr><th>Column</th><th>Description</th></tr>
<tr><td>Start Time</td><td>Time when the snapshot job started execution.

Click › to display details about the snapshot. For example:


Click the <u>View</u> link to open the Managed scheduled snapshots Command page, which displays details and messages about each step in the execution of the command. For example:

</td></tr>
<tr><td>Outcome</td><td>Displays whether the snapshot succeeded or failed.</td></tr>
<tr><td>Paths | Tables Processed</td><td>HDFS snapshots: the number of Paths Processed for the snapshot.

HBase snapshots: the number of Tables Processed for the snapshot.</td></tr>
<tr><td>Paths | Tables Unprocessed</td><td>HDFS Snapshots: the number of Paths Unprocessed for the snapshot.

HBase Snapshots: the number of Tables Unprocessed for the snapshot.</td></tr>
<tr><td>Snapshots Created</td><td>Number of snapshots created.</td></tr>
<tr><td>Snapshots Deleted</td><td>Number of snapshots deleted.</td></tr>
<tr><td>Errors During Creation</td><td>Displays a list of errors that occurred when creating the snapshot. Each error shows the related path and the error message.</td></tr>
<tr><td>Errors During Deletion</td><td>Displays a list of errors that occurred when deleting the snapshot. Each error shows the related path and the error message.</td></tr>
</table>

See Managing HDFS Snapshots on page 530 and Managing HBase Snapshots on page 519 for more information about managing snapshots.

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 removed from the policy may leave "orphaned" snapshots behind that are not deleted automatically because they are no longer associated with a current snapshot policy. Cloudera Manager never selects these snapshots for automatic deletion because selection for deletion only occurs when the policy creates a *new* snapshot containing those files, directories, or tables.

You can delete snapshots manually through Cloudera Manager or by creating a command-line script that uses the HDFS or HBase snapshot commands. Orphaned snapshots can be hard to locate for manual deletion. Snapshot policies automatically receive the prefix `cm-auto` followed by a globally unique identifier (GUID). You can locate all snapshots for a specific policy by searching for t the prefix `cm-auto-`*`guid`* that is unique to that policy.

To avoid orphaned snapshots, delete snapshots before editing or deleting the associated snapshot policy, or record 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. Recording the snapshot names, including the associated policy prefix, is necessary because the prefix associated with a policy cannot be determined after the policy has been deleted, and snapshot names do not contain recognizable references to snapshot policies.

Managing HBase Snapshots

This page demonstrates how to manage HBase snapshots using either Cloudera Manager or the command line.

Managing HBase Snapshots Using Cloudera Manager

For HBase services, you can use the Table Browser tab to view the HBase tables associated with a service on your cluster. You can view the currently saved snapshots for your tables, and delete or restore them. From the HBase Table Browser tab, you can:

- View the HBase tables for which you can take snapshots.
- Initiate immediate (unscheduled) snapshots of a table.
- View the list of saved snapshots currently maintained. These can 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

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. 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**)

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**.

   > **Important:** If AWS S3 access keys are rotated, the Cloudera Manager server must be restarted.

7. Enter the path to the location in Amazon S3 where your HBase snapshots will be stored in the field **Amazon S3 Path for Remote Snapshots**.

   > **Warning:** Do not use the Amazon S3 location defined by the path entered in **Amazon S3 Path for Remote Snapshots** for any other purpose, or directly add or delete content there. Doing so risks 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`. For 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

Starting with Cloudera Manager 5.8, YARN should by default allow the `hbase` user to run MapReduce jobs even when Kerberos is enabled. However, this change only applies to new Cloudera Manager deployments, and not if you have upgraded from a previous version to Cloudera Manager 5.8 (or higher).

If Kerberos is enabled on your cluster, and YARN does not allow the `hbase` user to run MapReduce jobs, 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 display 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 **Destination** section shows a choice of **Local** or **Remote S3**. Select **Remote S3**.
6. Click **Take Snapshot**.

   While the **Take Snapshot** command is running, a local copy of the snapshot with a name beginning `cm-tmp` followed by an auto-generated filename is displayed in the Table Browser. 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 might not be deleted. This copy can be manually deleted or kept as a valid local snapshot. To store a current snapshot in Amazon S3, run 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 creates a local copy of the remote snapshot with a name beginning with `cm-tmp` followed by an auto-generated filename, and uses that local copy to restore the table in HBase. Cloudera Manager then automatically deletes the local copy. If the **Restore** command fails without completing, the temporary copy might not be deleted and can be seen in the Table Browser. In that case, delete the local temporary copy manually and re-run the **Restore** command to restore the table from Amazon S3.

Restoring an HBase Snapshot from Amazon S3 with a New Name

By restoring an HBase snapshot stored in Amazon S3 with a new name, you clone 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 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. For snapshots stored in Amazon S3, you must also 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 and cannot change the setting later. If the setting is wrong, create a new policy.

When you create a snapshot based on a snapshot policy, a local copy of the snapshot is created with a name beginning with `cm-auto` followed by an auto-generated filename. The temporary copy of the snapshot is displayed in the Table Browser and 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 might not be deleted. 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.

Managing HBase Snapshots Using the Command Line

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

About HBase Snapshots

In previous HBase releases, the only way to a back up or to clone a table was to use `CopyTable` or `ExportTable`, or to copy all the `hfiles` in HDFS after disabling the table. These methods have disadvantages:

- `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 does 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 management tool; HBase does not have this functionality.

  - The database administrator may want to save a snapshot before a major application upgrade or change.

    > **Note:**
    >
    > Snapshots are not primarily used for system upgrade protection because they do not roll back binaries, and would not necessarily prevent bugs or errors in the system or the upgrade.

  - Recovery cases:
    - Roll back 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, and archiving.
  - Export from a snapshot on a live system provides a more consistent view of HBase than `CopyTable` and `ExportTable`.
- Audit or report view of data at a specific time
  - Capture monthly data for compliance.
  - Use for end-of-day/month/quarter reports.
- Application testing
  - Test schema or application changes on similar production data from a snapshot and then discard.

    For example:

    1. Take a snapshot.
    2. Create a new table from the snapshot content (schema and data)
    3. 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

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 required by the snapshot are in the /hbase/data/<namespace>/<tableName>/<regionName>/<familyName>/ location if the table is still using them; otherwise, they are 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.

To clone a table to another cluster, you export the snapshot to the other cluster and then run the clone operation; see Exporting a Snapshot to Another Cluster.

Reverting to a Previous HBase Version

Snapshots do not affect HBase backward compatibility if they are not used.

If you use snapshots, backward compatibility is affected as follows:

- If you only take snapshots, you can still revert to a previous HBase version.
- If you use restore or clone, you cannot revert to a previous version unless the cloned or restored tables have no links. Links cannot be detected automatically; you would need to inspect the file system manually.

Storage Considerations

Because hfiles are immutable, a snapshot consists of a reference to the files 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, butfor tables with lots of updates, and compactions, each snapshot could have a different set of hfiles.

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 do not affect HBase performance if they are not used.

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 `tableX` called `snapshotX`	`snapshot 'tableX', 'snapshotX'`	Snapshots can be taken while a table is disabled, or while a table is online and serving traffic. • If a table is disabled (using `disable <table>`), 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 `CopyTable` and is a significant improvement.
Restore snapshot `snapshotX` (replaces the source table content)	`restore_snapshot ‘snapshotX’`	For emergency use only; see Restrictions. Restoring a snapshot replaces the current version of a table with different version. To run this command, you must disable the target table. The `restore` 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. **Warning:** If you use coprocessors, the coprocessor must be available on the destination cluster before restoring the snapshot.
List all available snapshots	`list_snapshots`	
List all available snapshots starting with `‘mysnapshot_’` (regular expression)	`list_snapshots ‘my_snapshot_.*’`	
Remove a snapshot called `snapshotX`	`delete_snapshot ‘snapshotX’`	
Create a new table `tableY` from a snapshot `snapshotX`	`clone_snapshot ‘snapshotX’, ‘tableY’`	Cloning a snapshot creates a new read/write table that serves the data kept at the time of the snapshot. The original table and the cloned table can be modified independently; new data written to one table does not show up on the other.

Taking a Snapshot Using a Shell Script

You can take a snapshot using an operating system shell script, such as a Bash script, in HBase Shell noninteractive mode, which is described in Accessing HBase by using the HBase Shell on page 106. This example Bash script shows how to take a snapshot in this way. This script is provided as an illustration only; do not use in production.

```
#!/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 &lt;table&gt;"
 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 successA non-zero exit code indicates the possibility of failure, not a definite failure. Your script should check to see if the snapshot was created before taking the snapshot 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 524.

> **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, use the `-chuser`, `-chgroup`, or `-chmod` options as in the second example below, or be sure the destination directory has the correct permissions. In the following examples, replace the HDFS server path and port with the appropriate ones for your cluster.

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 can 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. 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 `Snapinfo` tool (see Information and Debugging on page 529) 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, and 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 require RegionServers to be up; otherwise, the snapshot fails.

> **Note:** This restriction also applies to a rolling upgrade, which can be done only through 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 exactly which rows you have lost), you can create a clone from the snapshot and create a MapReduce job to copy the data that you have lost.
>
> In this case, you do not 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 can 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.

```
$ 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
s3a://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 topic demonstrates how to manage HDFS snapshots using either Cloudera Manager or the command line.

Managing HDFS Snapshots Using Cloudera Manager

For HDFS services, use the File Browser tab to view the HDFS directories associated with a service on your cluster. You can view the currently saved snapshots for your files, and delete or restore them. 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 HDFS directory.
- View the list of saved snapshots currently being maintained. These can 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).

Before using snapshots, note the following limitations:

- Snapshots that include encrypted directories cannot be restored outside of the zone within which they were created.
- The Cloudera Manager Admin Console cannot perform snapshot operations (such as create, restore, and delete) for HDFS paths with encryption-at-rest enabled. This limitation only affects the Cloudera Manager Admin Console and does not affect CDH command-line tools or actions not performed by the Admin Console, such as BDR replication which uses command-line tools. For more information about snapshot operations, see the Apache HDFS snapshots documentation.

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).

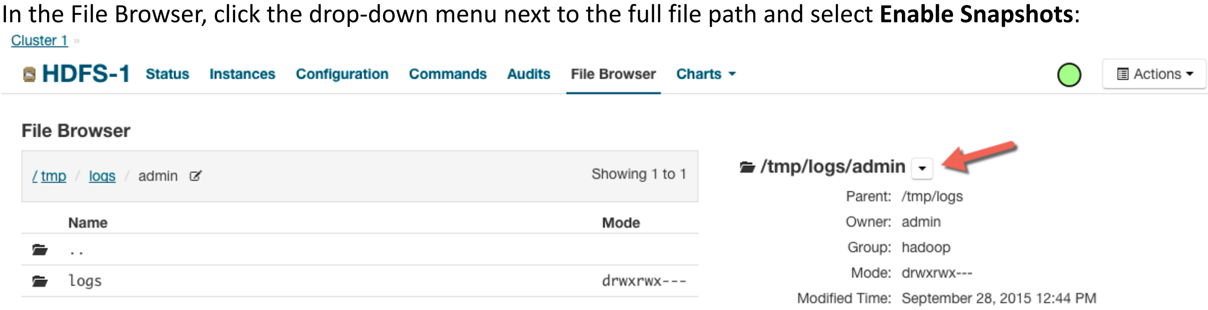
Enabling and Disabling HDFS Snapshots

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

For snapshots to be created, HDFS directories must be enabled for snapshots. You cannot specify a directory as part of a snapshot policy unless it has been enabled for snapshots.

Enabling an HDFS Directory for Snapshots

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.
3. Go to the directory you want to enable for snapshots.
4. In the File Browser, click the drop-down menu next to the full file path and select **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.

Disabling a Snapshottable Directory

To disable snapshots for a directory that has snapshots enabled, use **Disable Snapshots** from the drop-down menu button at the upper right. If snapshots of the directory exist, they must be deleted before snapshots can be disabled.

Taking and Deleting HDFS Snapshots

Minimum Required Role: Full Administrator

To manage HDFS snapshots, first enable an HDFS directory for snapshots.

Taking Snapshots

> **Note:** You can also schedule snapshots to occur regularly by creating a Snapshot Policy.

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.
3. Go to the directory with the snapshot you want to restore.
4. Click the drop-down menu next to the full path name and select **Take Snapshot**.

   The **Take Snapshot** screen displays.
5. Enter a name for the snapshot.
6. Click **OK**.

   The **Take Snapshot** button is present, enabling an immediate snapshot of the directory.
7. 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.

   Any snapshots that have been taken are listed by the time at which they were taken, along with their names and a menu button.

Deleting Snapshots

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.
3. Go to the directory with the snapshot you want to delete.
4. In the list of snapshots, locate the snapshot you want to delete and click .
5. Select **Delete**.

Restoring Snapshots

1. From the **Clusters** tab, select your CDH 5 HDFS service.
2. Go to the **File Browser** tab.
3. Go to the directory you want to restore.
4. In the File Browser, click the drop-down menu next to the full file path (to the right of the file browser listings) and select one of the following:
   - **Restore Directory From Snapshot**
   - **Restore Directory From Snapshot As...**

   The **Restore Snapshot** screen displays.
5. If you selected **Restore Directory From Snapshot As...**, enter the username to apply when restoring the snapshot.
6. Select one of the following:
   - **Use HDFS 'copy' command** - This option executes more slowly and does not require credentials in a secure cluster. It copies the contents of the snapshot as a subdirectory or as files within the target directory.

- **Use DistCp / MapReduce** - This options executes more quickly and requires credentials (Run As) in secure clusters. It merges the target directory with the contents of the source snapshot. When you select this option, the following additional fields, which are similar to those available when configuring a replication, display under **More Options**:

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

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

  - Enter the user to run the replication job in the **Run As Username** field. By default this is `hdfs`. If you want to run the job as a different user, enter the user name here. If you are using Kerberos, you *must* provide a user name here, and it must be one with an ID greater than 1000. (You can also configure the minimum user ID number with the **min.user.id** property in the YARN or MapReduce service.) Verify that the user running the job has a home directory, `/user/`*`username`*, owned by *`username`*`:supergroup` in HDFS. This user must have permissions to read from the source directory and write to the destination directory.

    Note the following:

    - The User must not be present in the list of banned users specified with the **Banned System Users** property in the YARN configuration (Go to the YARN service, select **Configuration** tab and search for the property). For security purposes, the `hdfs` user is banned by default from running YARN containers.
    - The requirement for a user ID that is greater than 1000 can be overridden by adding the user to the "white list" of users that is specified with the **Allowed System Users** property. (Go to the YARN service, select **Configuration** tab and search for the property.)

  - **Log path** - An alternate path for the logs.
  - **Maximum Map Slots** - Limits for the number of map slots per mapper. The default value is 20.
  - **Abort on Error** - Whether to abort the job on an error. If selected, files copied up to that point remain on the destination, but no additional files are copied. **Abort on Error** is off by default.
  - **Skip Checksum Checks** - Whether to skip checksum checks (the default is to perform them). If checked, checksum validation will not be performed.

    You must select the this property to prevent failure when restoring snapshots in the following cases:

    - Restoring a snapshot within a single encryption zone.
    - Restoring a snapshot from one encryption zone to a different encryption zone.
    - Restoring a snapshot from an unencrypted zone to an encrypted zone.

    See HDFS Transparent Encryption.

  - **Delete Policy** - Whether files that were deleted on the source should also be deleted from the destination directory. This policy also determines the handling of files in the destination location that are unrelated to the source. Options include:

    – **Keep Deleted Files** - Retains the destination files even when they no longer exist at the source. (This is the default.).
    – **Delete to Trash** - If the HDFS trash is enabled, files are moved to the trash folder. (Not supported when replicating to Amazon S3.)
    – **Delete Permanently** - Uses the least amount of space; use with caution.

  - **Preserve** - Whether to preserve the block size, replication count, permissions (including ACLs), and extended attributes (XAttrs) as they exist on the source file system, or to use the settings as configured on the destination file system. By default source system settings are preserved. When **Permission** is checked, and both the source and destination clusters support ACLs, replication preserves ACLs. Otherwise, ACLs are not replicated. When **Extended attributes** is checked, and both the source and destination clusters support extended attributes, replication preserves them. (This option only displays when both source and destination clusters support extended attributes.)

If you select one or more of the **Preserve** options and you are replicating *to* Amazon S3, the values all of these items are saved in meta data files on S3. When you replicate *from* Amazon S3 to HDFS, you can select which of these options you want to preserve.

> **Note:** To preserve permissions to HDFS, you must be running as a superuser on the *destination* cluster. Use the "Run As Username" option to ensure that is the case.

Managing HDFS Snapshots Using the Command Line

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

For information about managing snapshots using the command line, see HDFS Snapshots.

BDR Tutorials

Cloudera Backup and Disaster Recovery (BDR) is available with a Cloudera Enterprise license. Enterprise BDR lets you replicate data from one cluster to another, or from one directory path to another on the same or on a different cluster. In case of data loss, the backup replica can be used to restore data to the production cluster.

The time to start thinking about how to restore data is long before you might ever need to do so. These BDR tutorials take you step-by-step through the process of backing up an example production cluster. The example backup replication schedules are for one-time replication that makes a backup copy of Hive datasets or of HDFS files, respectively, on another cluster designated as a backup cluster.

The restore processes detailed in each tutorial also take you step-by-step through the process of restoring data using two different general approaches:

- How To Back Up and Restore Apache Hive Data Using Cloudera Enterprise BDR on page 534 highlights a *one-off* data recovery scenario in which you create the replication schedule immediately after a data loss and use it to restore data.
- How To Back Up and Restore HDFS Data Using Cloudera Enterprise BDR on page 546 shows you how pre-configure replication schedules so they are available when needed.

Use either or both of these tutorials to help plan your own backup and restore strategy.

How To Back Up and Restore Apache Hive Data Using Cloudera Enterprise BDR

Cloudera Enterprise Backup and Disaster Recovery (BDR) uses replication schedules to copy data from one cluster to another, enabling the second cluster to provide a backup for the first.

This tutorial shows you how to configure replication schedules to back up Apache Hive data and to restore data from the backup cluster when needed.

Creating replication schedules for backup and restore requires:

- A license for Cloudera Enterprise. Cloudera Enterprise BDR is available from the **Backup** menu of Cloudera Manager Admin Console when licensed for Enterprise.

- The **BDR Administrator** or **Full Administrator** role on the clusters involved (typically, a production cluster and a backup cluster).

Best Practices for Back Up and Restore

When configuring replication schedules for Hive back up and restore, follow these guidelines:

- Make sure that the time-frames configured for replication schedules allow the replication process to complete.
- Test your replication schedules for both back up and restore before relying on them in a production environment. To test a restore replication schedule in a production environment, use a different HDFS destination path for the Hive data files than that used for the replica.
- Enable only one replication schedule for the same dataset at the same time. That means you must first disable the backup replication schedule before enabling or creating a restore replication schedule for the same dataset, and vice versa.
- Enable snapshots on the HDFS file system containing the Hive data files. This ensures consistency if changes are still being made during the replication process. See Using Snapshots with Replication on page 502 for details.

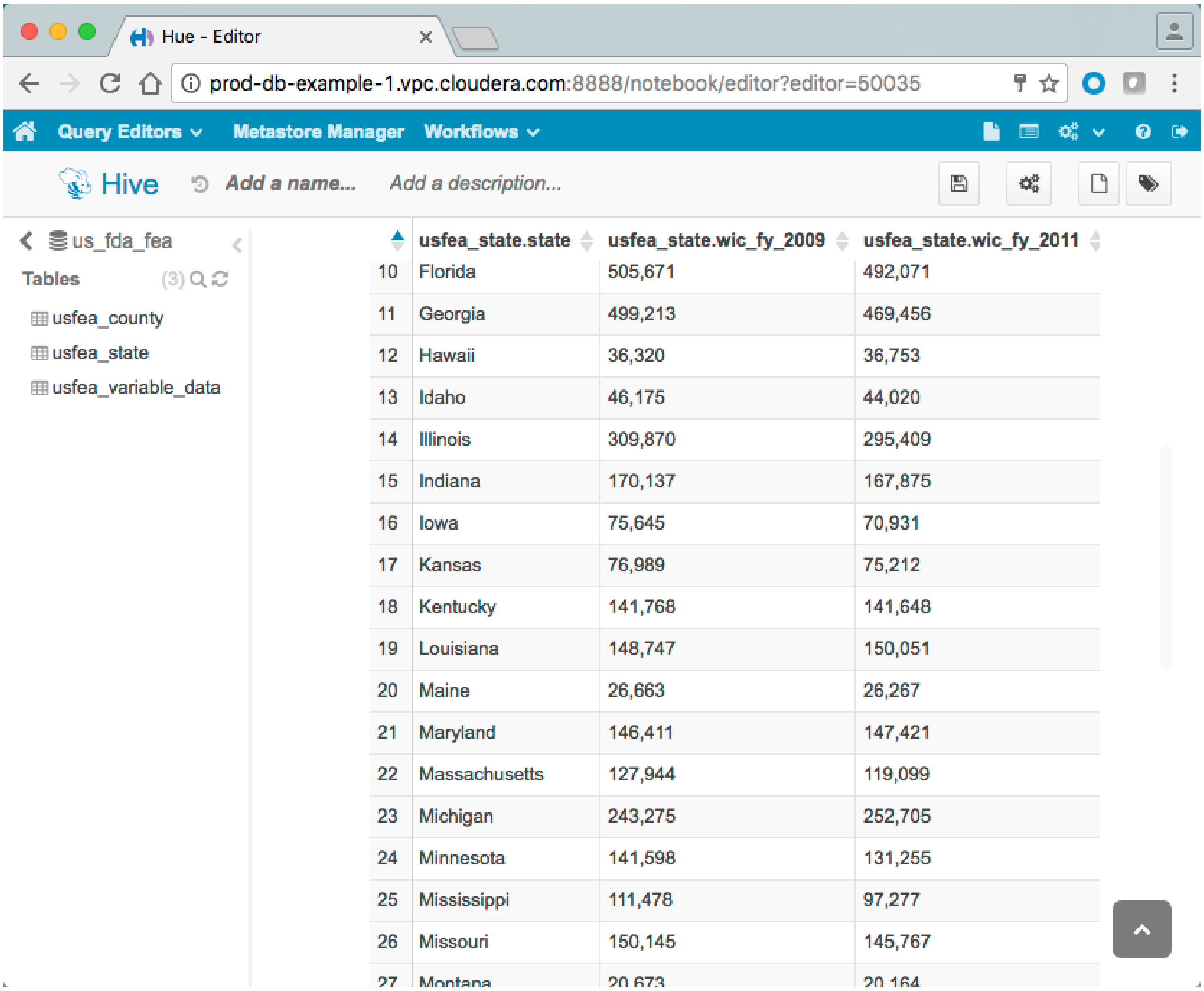
About the Example Clusters

This guide uses the two example clusters listed in the following table:

Production cluster	Backup cluster
http://prod-db-example-1.vpc.cloudera.com	http://backup-example-1.vpc.cloudera.com
Source cluster for a backup replication schedule.	Destination cluster for backup replication schedule.
Destination cluster for a restore replication schedule.	Source cluster for a restore replication schedule.
To restore data from a backup cluster, set peer relationship to backup cluster and configure replication schedule from this cluster.	To back up a production cluster, set peer relationship and configure replication schedule from this cluster.

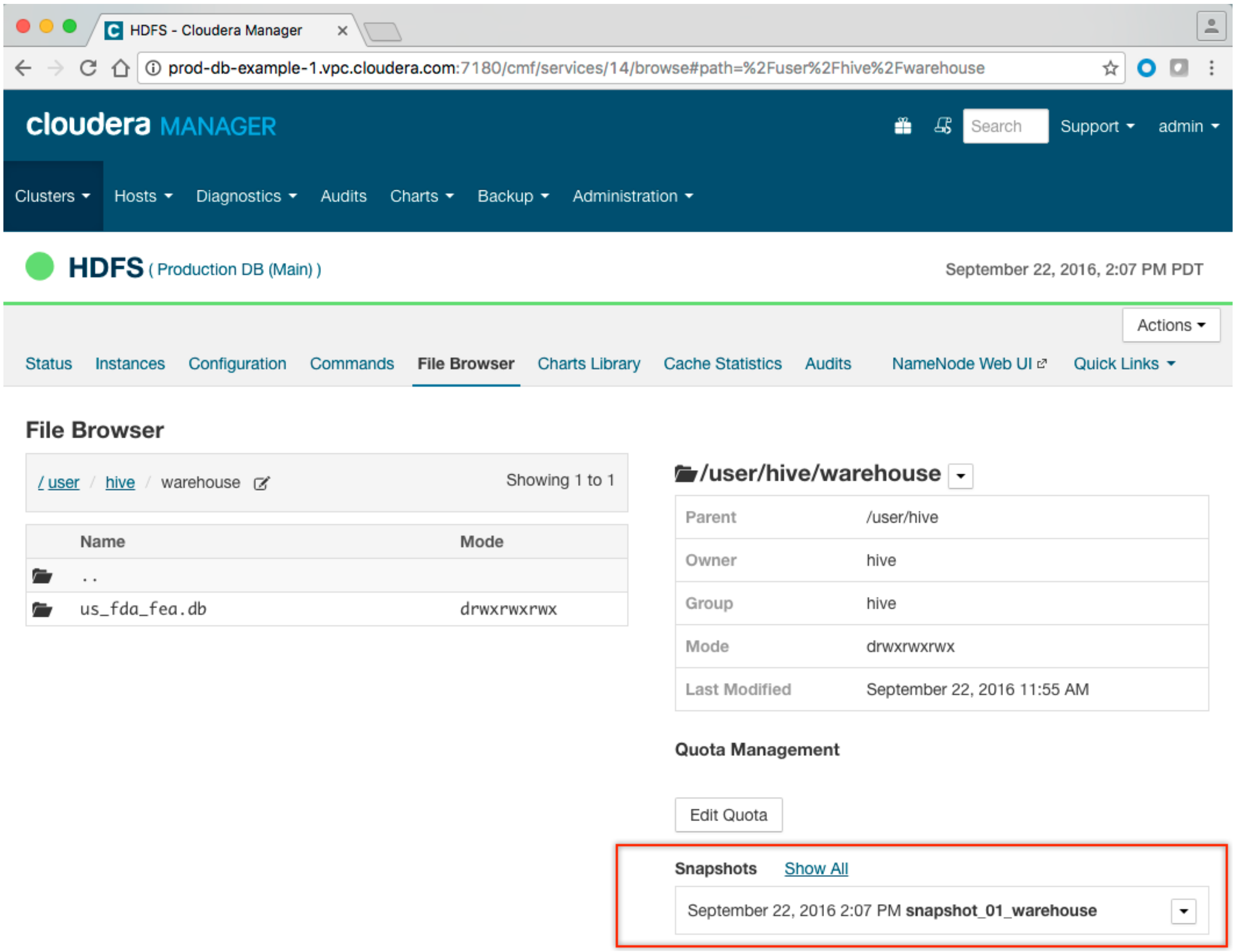
The example clusters are not configured to use Kerberos, nor do they use external accounts for cloud storage on Amazon Web Services (AWS). The name of the example production and the example backup cluster have each been changed from the default "Cluster 1" name to *Production DB (Main)* and *Offsite-Backup*, respectively.

The example production cluster contains the Hive *default* database and an example database, *us_fda_fea*, which contains data extracted from the US federal government's open data initiative at data.gov. The us_fda_fea database contains three tables as shown in Hue Web UI:

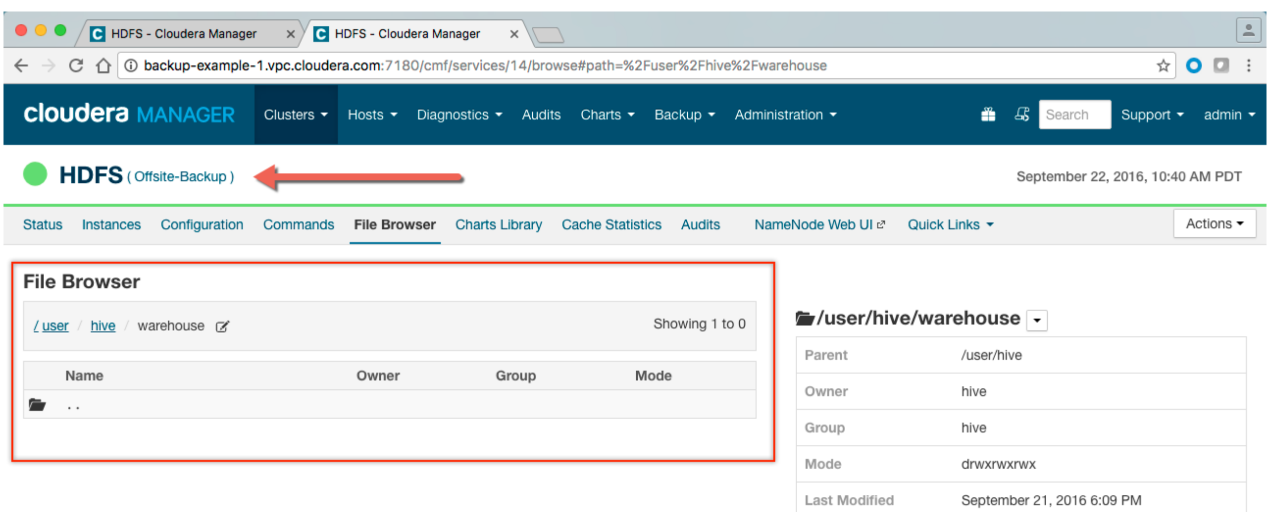


	usfea_state.state	usfea_state.wic_fy_2009	usfea_state.wic_fy_2011
10	Florida	505,671	492,071
11	Georgia	499,213	469,456
12	Hawaii	36,320	36,753
13	Idaho	46,175	44,020
14	Illinois	309,870	295,409
15	Indiana	170,137	167,875
16	Iowa	75,645	70,931
17	Kansas	76,989	75,212
18	Kentucky	141,768	141,648
19	Louisiana	148,747	150,051
20	Maine	26,663	26,267
21	Maryland	146,411	147,421
22	Massachusetts	127,944	119,099
23	Michigan	243,275	252,705
24	Minnesota	141,598	131,255
25	Mississippi	111,478	97,277
26	Missouri	150,145	145,767
27	Montana	20,673	20,164

As shown in the screenshot below, snapshots have been configured and enabled on the HDFS system path containing the Hive database files. Using snapshots is one of the Best Practices for Back Up and Restore on page 535. See Using Snapshots with Replication on page 502 for more information.



The example backup cluster (*Offsite-Backup*) has not yet been used for a backup yet, so the Hive default path is empty as shown below:

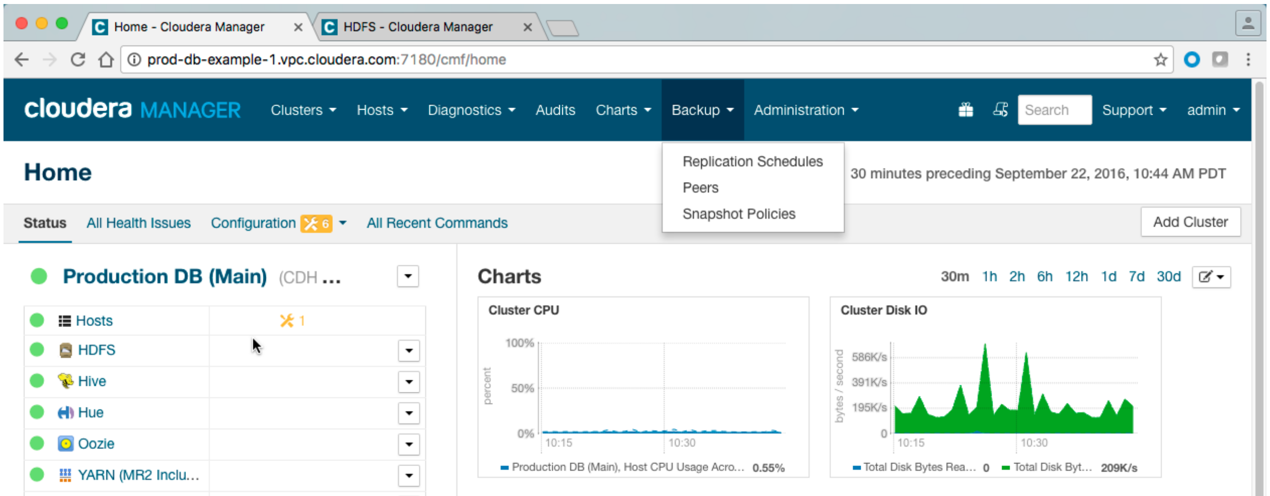


Backing Up and Restoring Hive Data

This tutorial steps through these two major tasks:

- Creating a Backup on page 538
- Restoring Data from the Backup Cluster on page 542

Backup and restore are each configured and managed using Replication Schedules, available from the Backup menu on Cloudera Manager Admin Console:



> **Note:** Screenshots in this guide show version 5.9 of the Cloudera Manager Admin Console.

The backup and restore processes are configured, managed, and executed using replication schedules. Each replication schedule identifies a *source* and a *destination* for the given replication process. The replication process uses a *pull* model. When the replication process runs, the configured destination cluster accesses the given source cluster and transparently performs all tasks needed to recreate the Hive database and tables on the destination cluster.

The destination cluster handles configuration and running the schedule. Typically, creating a backup replication schedule takes place on the backup cluster and creating a restore replication schedule takes place on the production cluster. Thus, as shown in this tutorial, the example production cluster, *Production DB (Main)*, is the *source* for the backup replication schedule and the *destination* for the restore replication schedule.

Creating a Backup

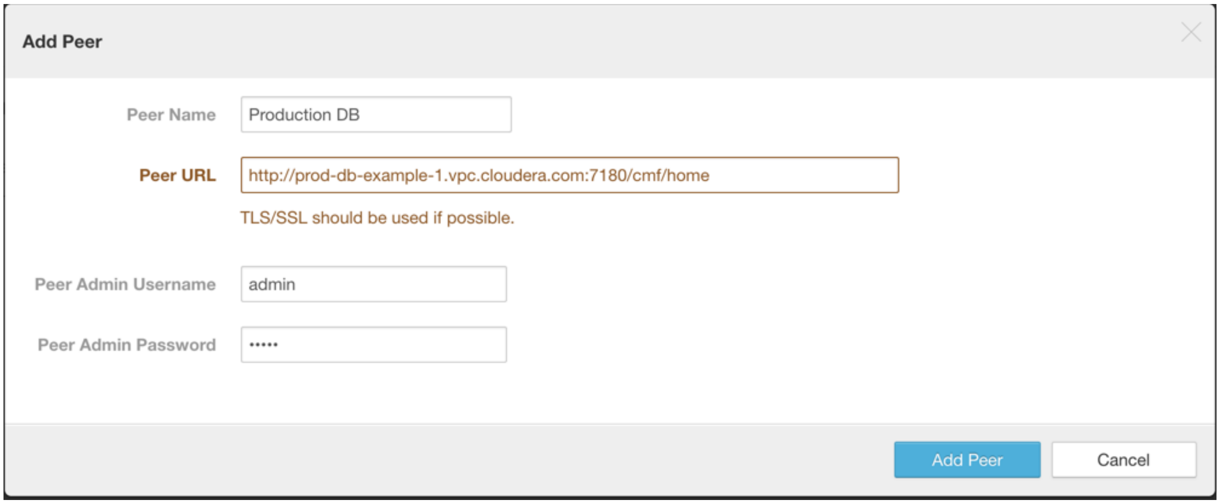
Defining the backup replication schedule starts from the Cloudera Manager Admin Console on the destination cluster. For this example, the *destination* cluster is the cluster being used as the backup and the *source* is the example production cluster. To create the backup, follow these steps:

- Step 1: Establish a Peer Relationship to the Production Cluster on page 538
- Step 2: Configure the Replication Schedule on page 539
- Step 3: Verify Successful Replication on page 540

Step 1: Establish a Peer Relationship to the Production Cluster

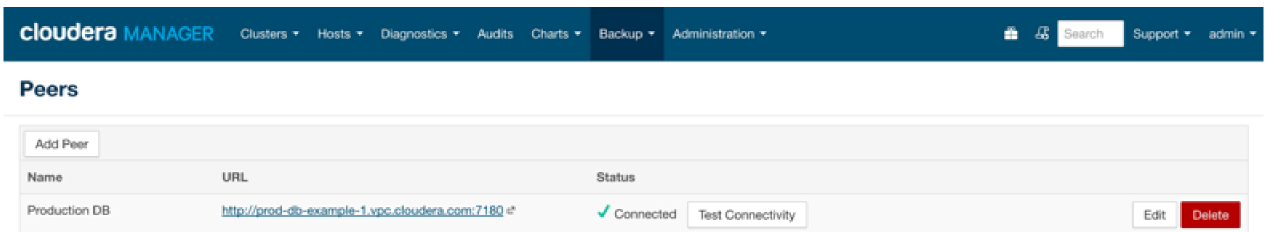
You must have the BDR Administrator or Full Administrator role on both clusters to define a Peer relationship and perform all subsequent steps.

1. Log in to Cloudera Manager Admin Console on the master node of the backup cluster.
2. Click the **Backup** tab and select **Peers** from the menu.
3. On the Peers page, click **Add Peers**.
4. On the Add Peer page:
    a. **Peer Name** - Enter a meaningful name for the cluster that you want to back up, such as *Production DB*. This peer name becomes available in the next step, to be selected as the source for the replication.
    b. **Peer URL** - Enter the URL (including port number) for the Cloudera Manager Admin Console running on the master node of the cluster.
    c. **Peer Admin Username** - Enter the name of the administrator account for the cluster.
    d. **Peer Admin Password** - Enter the password for the administrator account.



5. Click **Add Peer** to save your settings, connect to the production cluster, and establish this peer relationship.

The Peers page re-displays, showing the Status column as Connected (note the green check-mark):

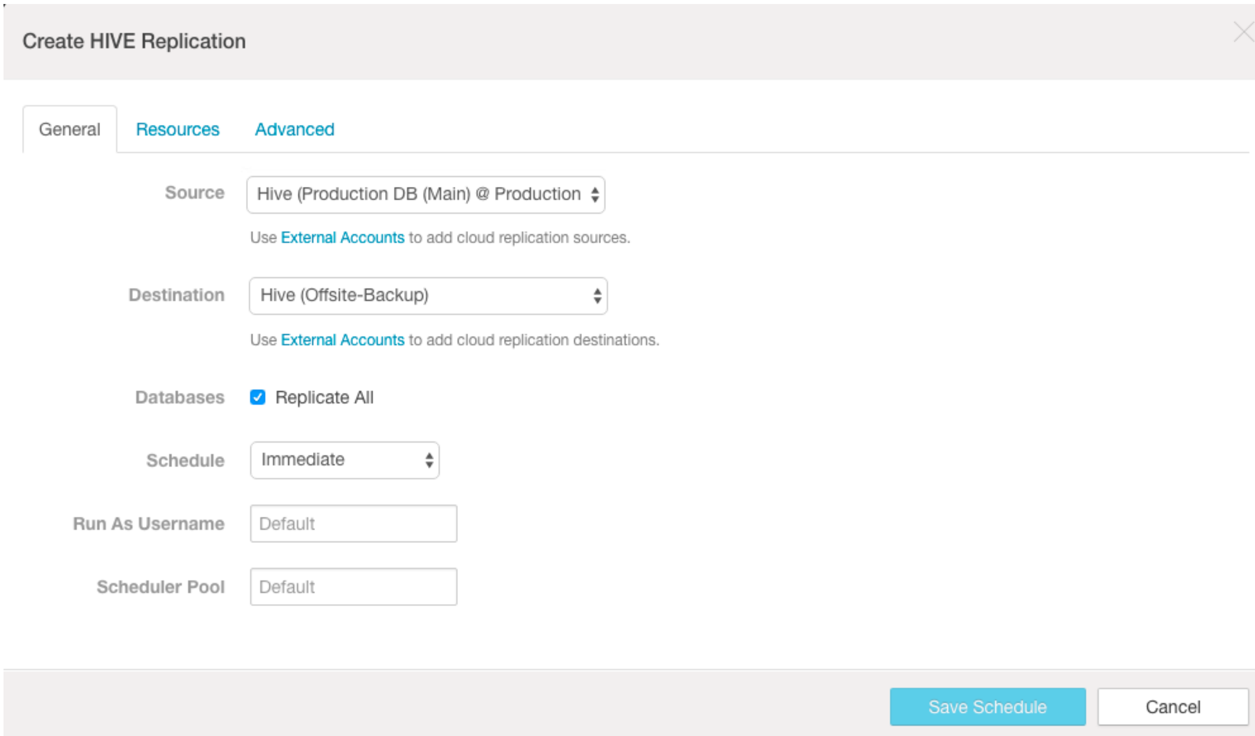


You can now create a schedule to replicate Hive files from production to the backup cluster.

Step 2: Configure the Replication Schedule

From the Cloudera Manager Admin Console on the backup cluster:

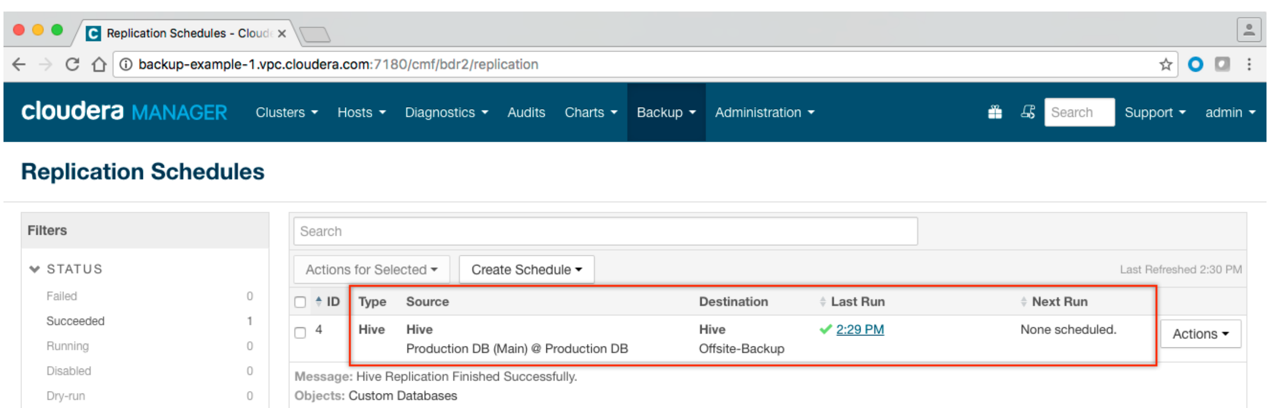
1. Click the **Backup** tab and select **Replication Schedules** from the menu.
2. On the Replication Schedules page, select **Create Schedule** > **Hive Replication**.
3. On the Create Hive Replication page, click the General tab to display the default schedule options:
    a. **Source** - Make sure the Hive node selected in the drop-down is the production cluster (the cluster to be backed up).
    b. **Destination** - This is the cluster you are logged into, the backup cluster. Select the Hive node on the cluster.
    c. **Databases** - Select Replicate All to re-create all Hive databases from the production system to the backup. Or deselect Replicate All and enter specific database name and tables to back up select databases or tables only.
    d. **Schedule** - Immediate. For production environments, change this to Recurring and set an appropriate time-frame that can backup the selected dataset completely. For example, do not set an hourly schedule if it takes two hours to back up the dataset.
    e. **Run As Username** - Leave as Default.
    f. **Scheduler Pool** - Leave as Default.
4. Click Save Schedule.



The files are replicated from the source cluster to the backup cluster immediately.

> **Note:** When you configure a replication schedule to back up Hive data on a regular basis, make sure that the schedule allows for each backup to complete. For example, do not create a schedule to back up every hour if it takes two hours to complete a full backup.

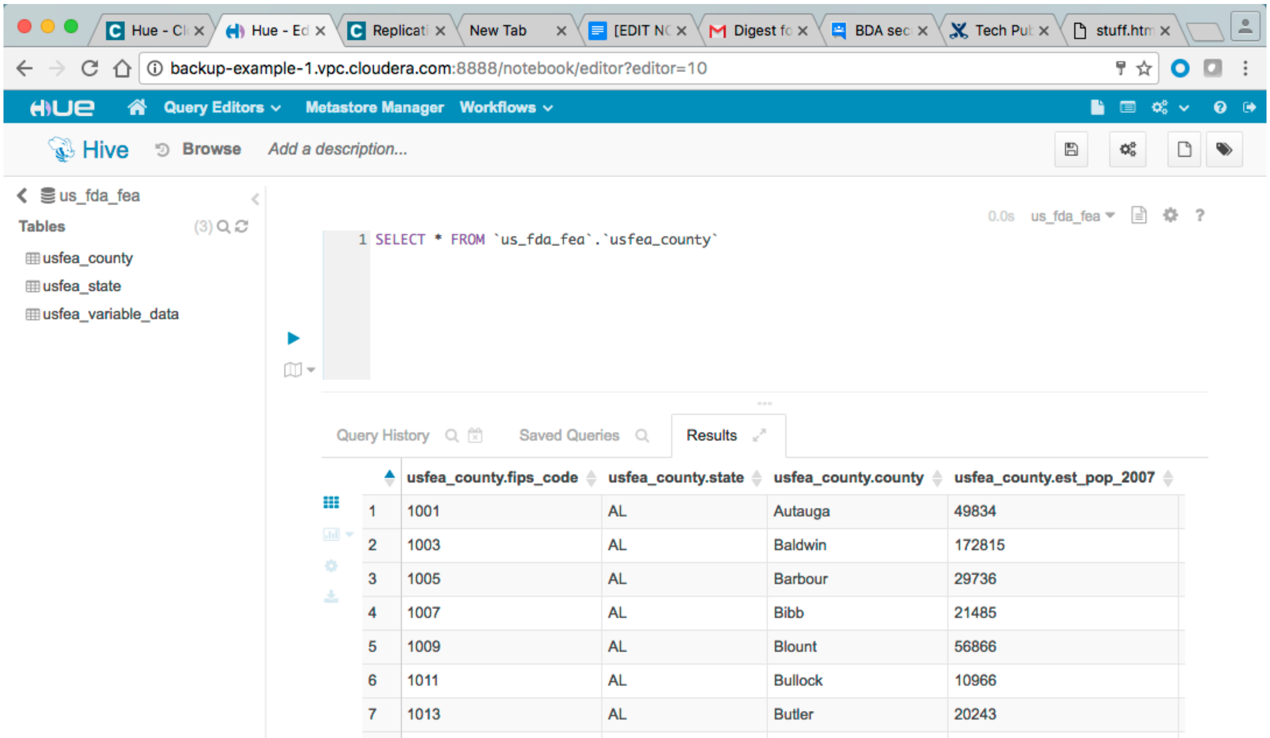
When the process completes, the Replication Schedules page re-displays, showing a green check-mark and time-stamp in the Last Run column:



When you set up your own schedules in your actual production environment, the Next Run column will likely also contain a date and time according to your specifications for the schedule.

Step 3: Verify Successful Replication

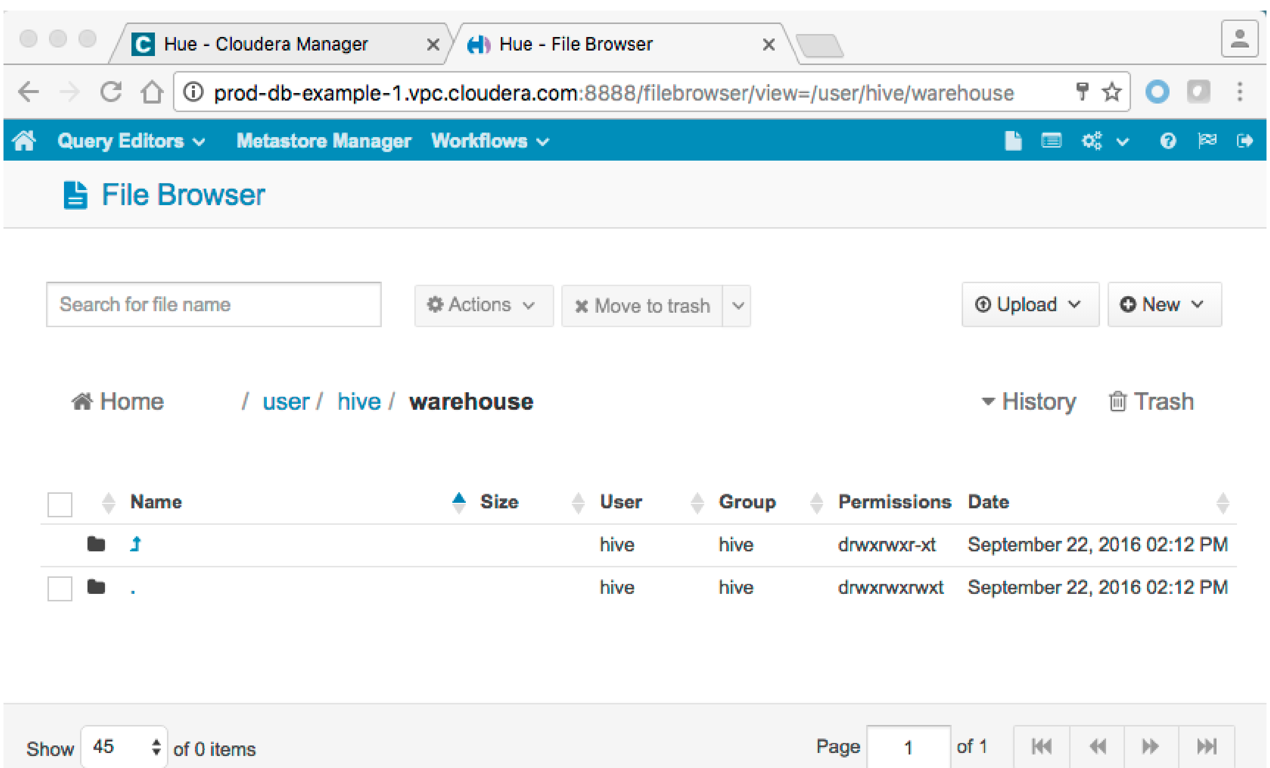
You can verify that data has been replicated by using Hive commands, the HDFS File Browser, or the Hue Web UI (shown below):



The Hive database is now on both the production and the backup clusters—the source and destination of the backup Replication Schedule, respectively.

At this point if the production cluster has a catastrophic data loss, you can use the backup replica to restore the database to the production cluster.

For example, assume that the us_fda_fea database was inadvertently deleted from the example production cluster as shown in the Hue Web UI:



Whenever you first discover an issue (data loss, corruption) with production data, *immediately disable* any existing backup replication schedules. Disabling the backup replication schedule prevents corrupt or missing data from being replicated over an existing backup replica is why it is the first step in the restore process detailed in the next set of steps.

Restoring Data from the Backup Cluster

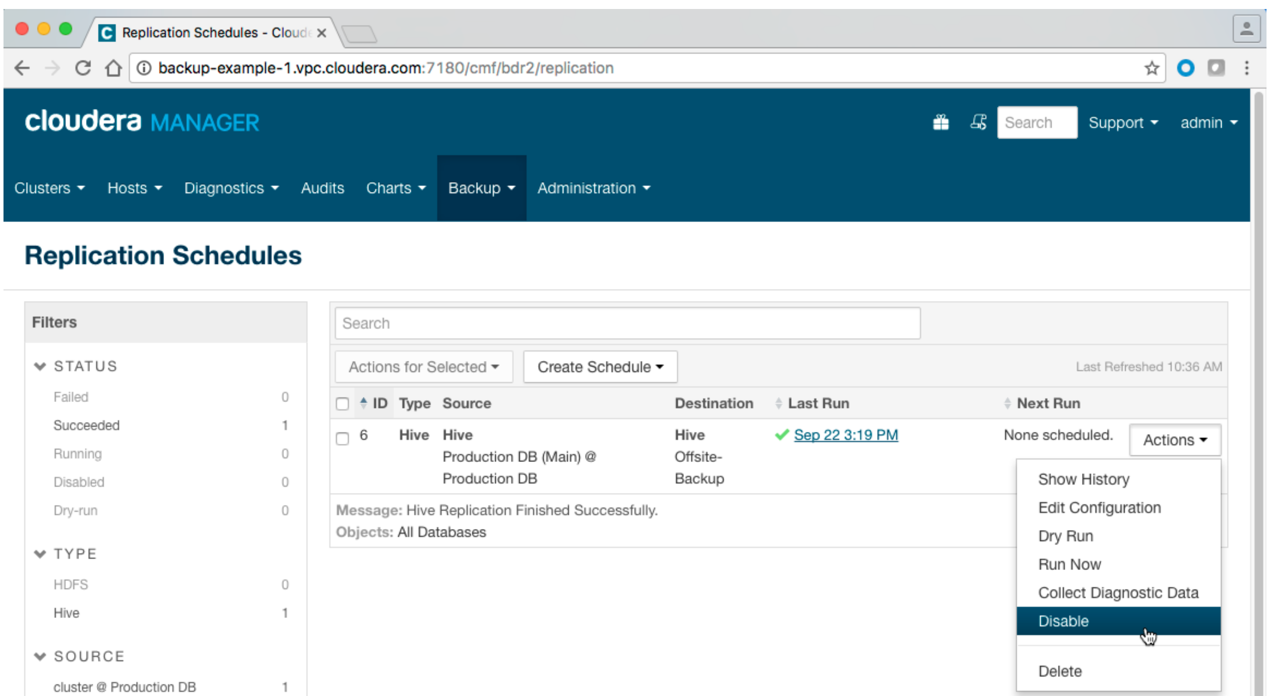
Restoring data from a backup cluster takes place on the production cluster but requires that the backup replication schedule is first disabled. The process includes these steps:

- Step 1: Disable Backup Replication Schedule on page 542
- Step 2: Establish a Peer Relationship to the Backup Cluster on page 542
- Step 3: Configure the Restore Replication Schedule on page 543
- Step 4: Disable the Replication Schedule on page 544
- Step 5: Verify Successful Replication on page 544
- Step 6: Re-enable the Backup Replication Schedule on page 545

Step 1: Disable Backup Replication Schedule

At the Cloudera Manager Admin Console on the backup cluster:

1. Select **Backup** > **Replication Schedules**.
2. On the **Replication Schedules** page, select the schedule.
3. From the **Actions** drop-down menu, select **Disable**.



When the Replication Schedules pages refreshes, the word Disabled displays in the Next Run column for the schedule.

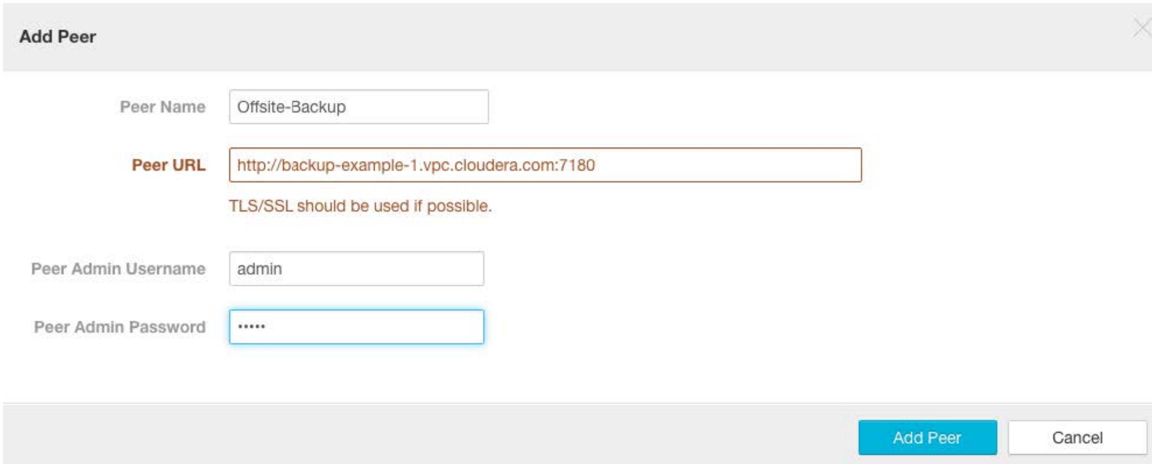
With the backup replication schedule temporarily disabled, move to the production cluster to create and run the replication schedule to restore the data as detailed in the remaining steps.

Step 2: Establish a Peer Relationship to the Backup Cluster

Log in to Cloudera Manager Admin Console on the master node of the production cluster.

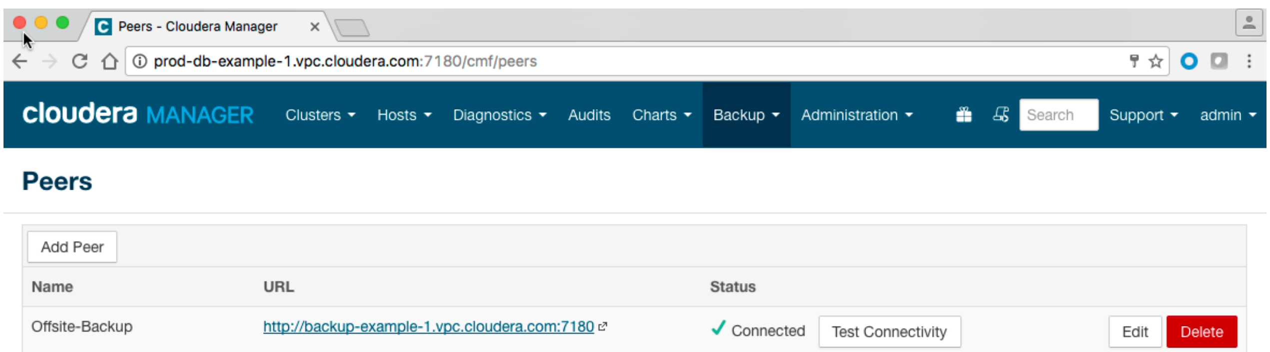
1. Click the **Backup** tab and select **Peers** from the menu.
2. On the **Peers** page, click **Add Peers** button.
3. On the **Add Peer** page:

   a. **Peer Name** - Enter a meaningful name for the cluster from which to obtain the backup data.
   b. **Peer URL** - Enter the URL for the Cloudera Manager Admin Console (running on the master node of the cluster).
   c. **Peer Admin Username** - Enter the administrator user name for the backup cluster.



4. Click **Add Peer** to save your settings. The production cluster connects to the backup cluster, establishes the peer relationship, and tests the connection.

The Peers page redisplays and lists the peer name, URL, and shows its Status (Connected) as shown below:



Step 3: Configure the Restore Replication Schedule

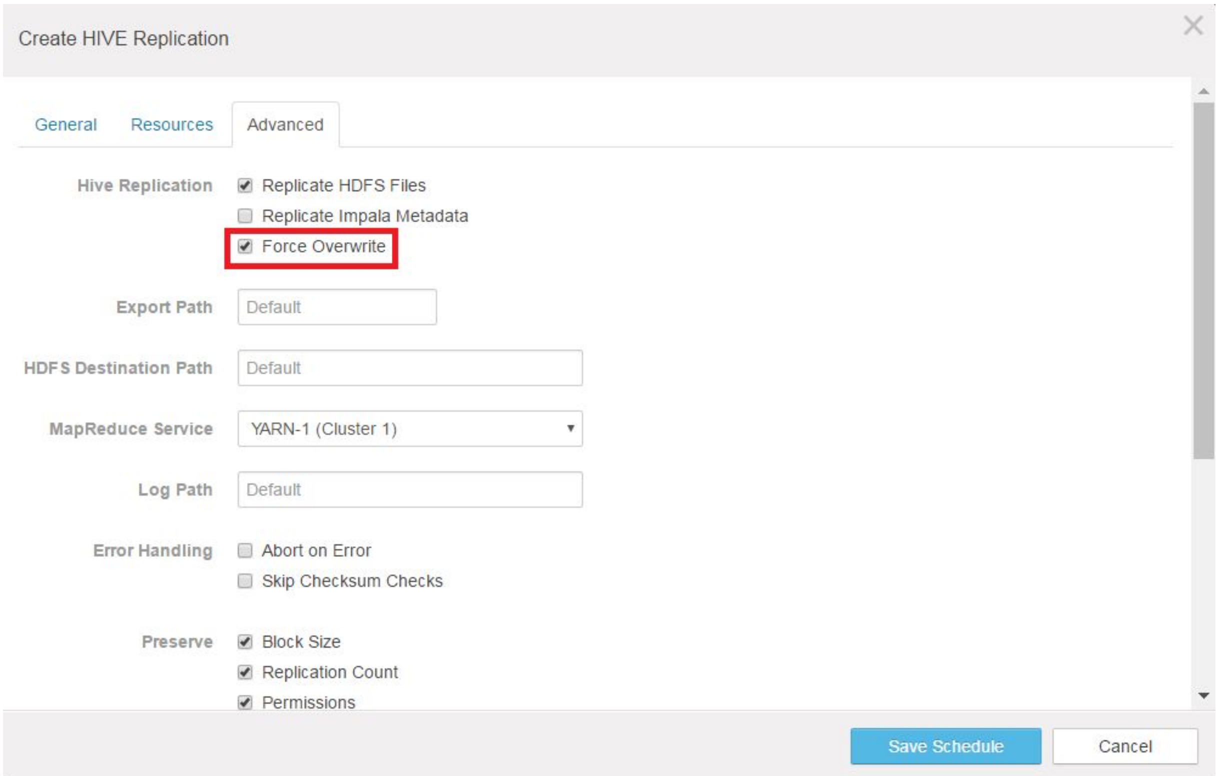
From the Cloudera Manager Admin Console on the production cluster:

1. Click the **Backup** tab and select **Replication Schedules** from the menu.
2. On the **Replication Schedules** page, select **Create Schedule** > **Hive Replication**.
3. On the General settings tab of the Create Hive Replication page:

   a. **Source** - The backup cluster from which to pull the data.
   b. **Destination** - The production cluster that needs the data restored.
   c. **Databases** - Select Replicate All.
   d. **Schedule** - Immediate.
   e. **Run As Username** - Leave as Default.
   f. **Scheduler Pool** - Leave as Default.

> **Important:** Be sure that the *source* is the cluster where your backup is stored and the *destination* is the cluster containing lost or damaged data that you want to replace with the backup.

4. Click the **Advanced** tab:



5. Select **Force Overwrite** so that the backup cluster's metadata replaces the metadata on the production cluster. The assumption is that the production cluster's dataset has been corrupted.

> **Important:** The Force Overwrite setting can destroy tables or entries created after the backup completed. Do not use this setting unless you are certain that you want to overwrite the destination cluster with data from the source.

6. Click **Save Schedule**.

This schedule runs one time and restores your Hive databases. When the process completes, the Replication Schedules page displays the time-stamp and check-mark in the Last Run column for the schedule.

Step 4: Disable the Replication Schedule

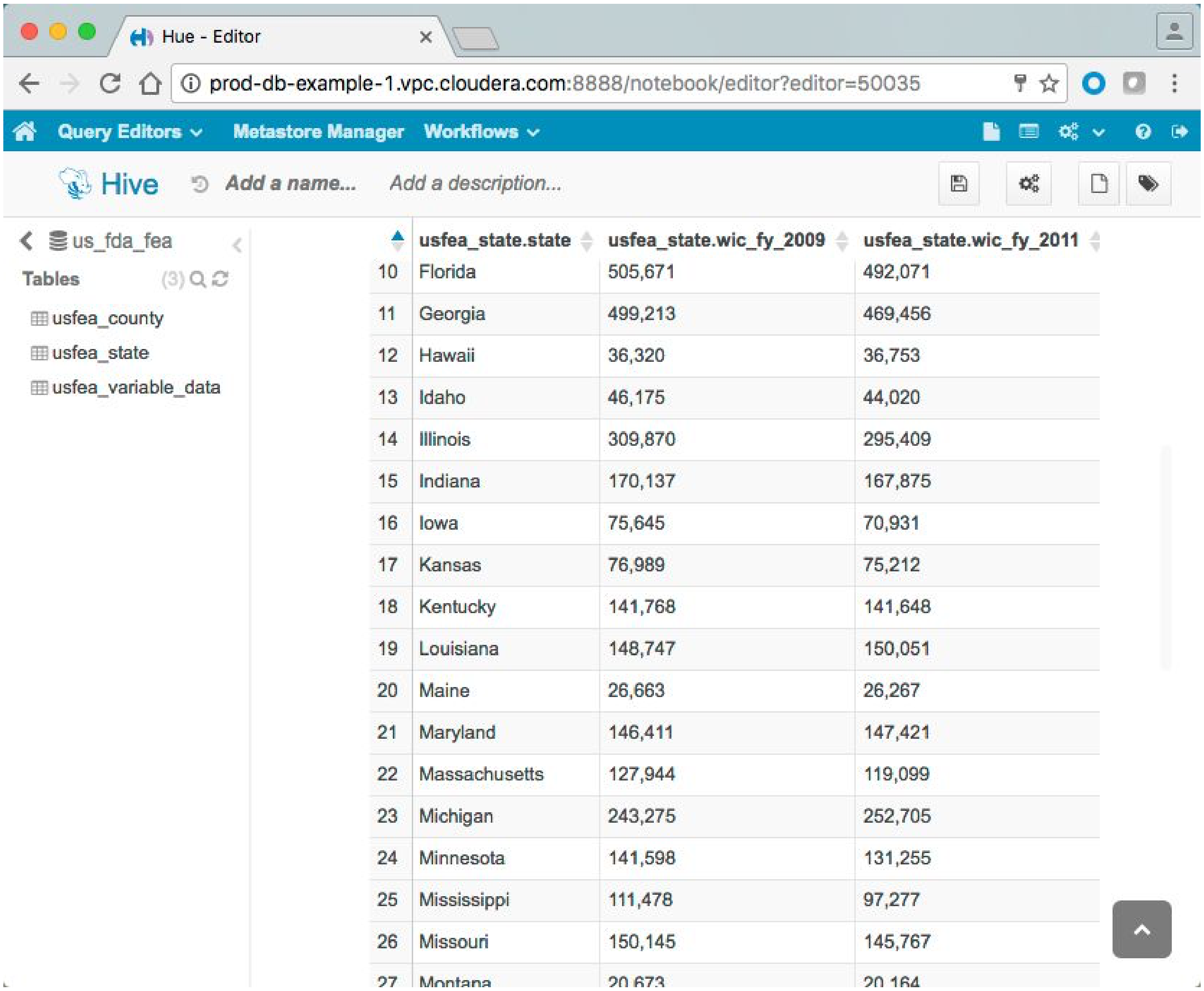
Immediately disable the Replication Schedule used for the restore as soon as it completes. From the Replication Schedules page:

1. Select the replication schedule that just completed.
2. From the **Actions** drop-down menu, select **Disable**.

The page refreshes and displays **Disabled** in the Next Run column.

Step 5: Verify Successful Replication

Use the Hive command-line, the HDFS File Browser, or the Hue Web UI to verify successful data restore:



The restore process is complete. In a production environment, assuming the restored Hive database and tables are as you want them in a temporary path, you can re-configure the replication schedule to restore the data to the original path.

Step 6: Re-enable the Backup Replication Schedule

On the backup cluster, log in to the Cloudera Manager Admin Console.

1. Select **Backup** > **Replication Schedules**.
2. On the Replication Schedules page, select the schedule created at the beginning of this process.
3. From the **Actions** drop-down menu, select **Enable**.

The restore process is complete.

In actual production environments, create replication schedules that regularly back up your production clusters. To restore data, create replication schedules as shown in this tutorial.

Alternatively, you can define replication schedules in advance but leave them disabled. See How To Back Up and Restore HDFS Data Using Cloudera Enterprise BDR on page 546 for details.

See Backup and Disaster Recovery on page 473 and BDR Tutorials on page 534 or more information about Cloudera Enterprise BDR.

How To Back Up and Restore HDFS Data Using Cloudera Enterprise BDR

Cloudera Enterprise Backup and Disaster Recovery (BDR) uses replication schedules to copy data from one cluster to another, enabling the second cluster to provide a backup for the first. In case of any data loss, the second cluster—the backup—can be used to restore data to production.

This tutorial shows you how to create a replication schedule to copy HDFS data from one cluster to another for a backup, and how to create and test a replication schedule that you can use to restore data when needed in the future.

Creating replication schedules for backup and restore requires:

- A license for Cloudera Enterprise. Cloudera Enterprise BDR is available from the **Backup** menu of Cloudera Manager Admin Console when licensed for Enterprise.
- The **BDR Administrator** or **Full Administrator** role on the clusters involved (typically, a production cluster and a backup cluster).

Best Practices for Back Up and Restore

When configuring replication schedules for HDFS back up and restore, follow these guidelines:

- Make sure that the time-frames configured for replication schedules allow for the replication process to complete.
- Create a restore replication schedule in advance but leave it disabled, as shown in this tutorial.
- Test your replication schedules for both back up and restore before relying on them in a production environment.
- Enable only one replication schedule for the same dataset at the same time. That means you must first disable the backup replication schedule before enabling or creating a restore replication schedule for the same dataset, and vice versa.
- Enable snapshots on the HDFS file system. Snapshots ensure consistency if changes are still being made to data during the replication process. See Using Snapshots with Replication on page 502 for details.

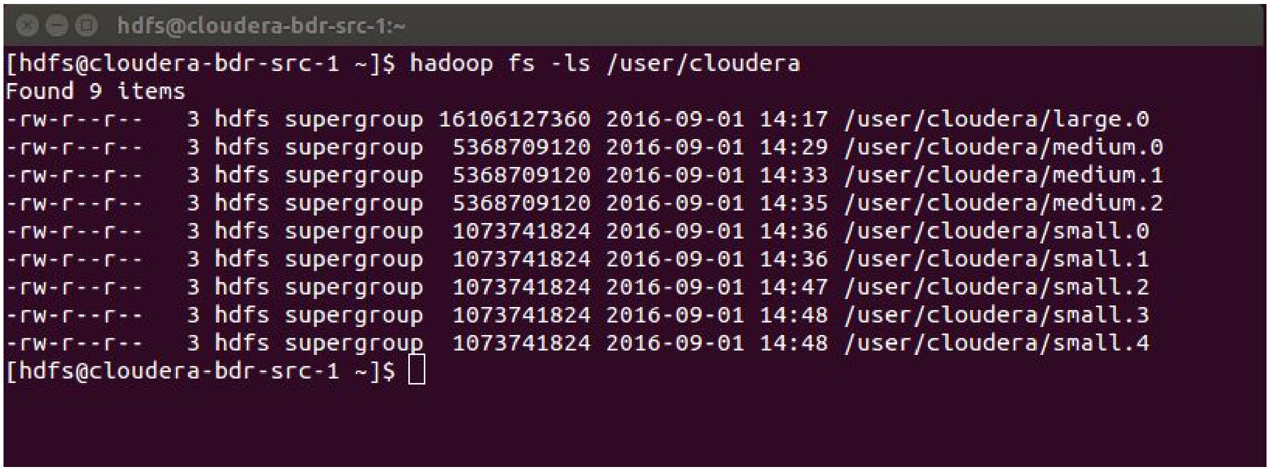
About the Example Clusters

This tutorial uses the two example clusters listed in the table. The nodes shown are the master nodes for the clusters.

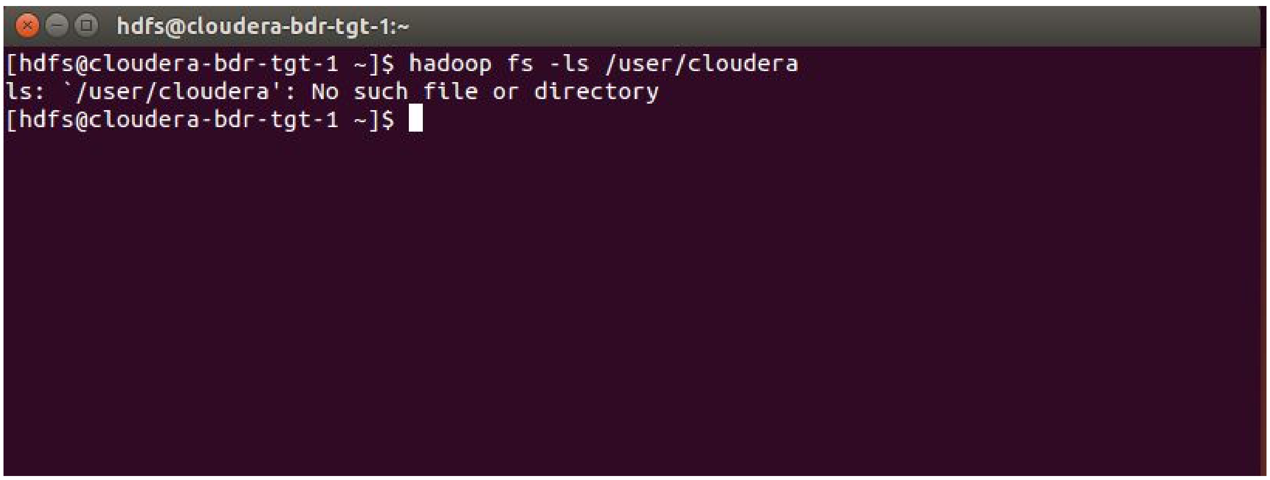
Production cluster	Backup cluster
cloudera-bdr-src-{1..4}.cloud.computers.com	cloudera-bdr-tgt-{1..4}.cloud.computers.com
http://cloudera-bdr-src-1.cloud.computers.com	http://cloudera-bdr-tgt-1.cloud.computers.com
Source cluster for a backup replication schedule.	Destination cluster for a backup replication schedule.
Destination for a restore replication schedule.	Source for a restore replication schedule.
For restore, set peer relationship from this cluster.	For backup, set peer relationship from this cluster. To create an initial backup, set Schedule to Immediate.

The example clusters are not configured to use Kerberos, nor do they use external accounts for cloud storage on Amazon Web Services (AWS).

The example production cluster contains nine HDFS files in the `/user/cloudera` path:



The example backup cluster has not been used as the destination of a replication schedule yet, so the HDFS file system has no `/user/cloudera` directory:



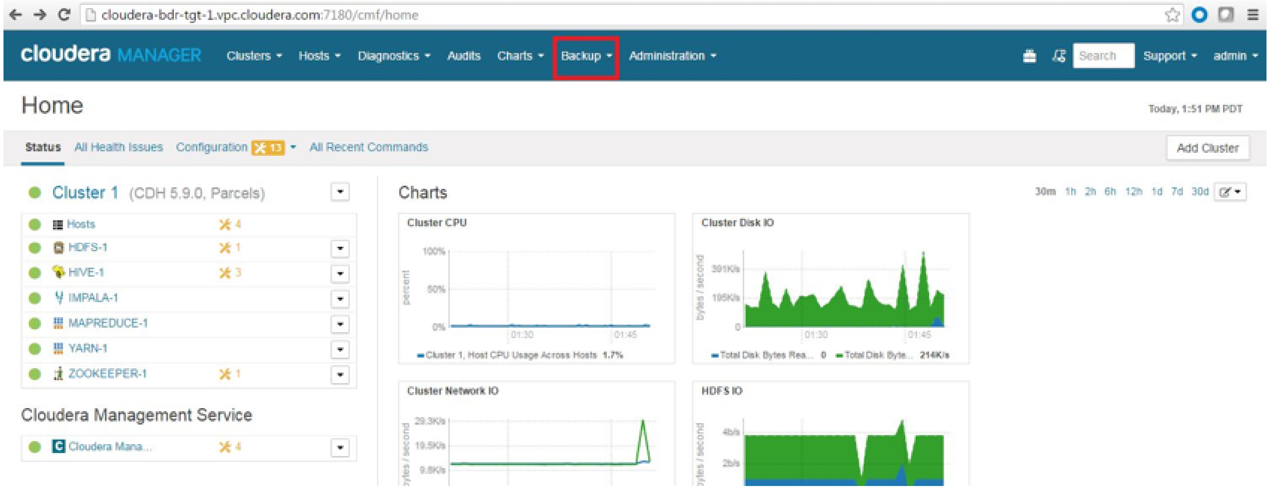
Backing Up and Restoring HDFS Data

This tutorial steps through the processes of creating and running a backup replication schedule, and creating a restore replication schedule designed for future use.

Preparing for disaster recovery includes these three major tasks:

- Backing Up HDFS Files on page 548
- Configuring a Restore Replication Schedule on page 551
- Recovering from Catastrophic Data Loss on page 555

Backup and restore are both configured and managed using Replication Schedules, available from the Backup menu on Cloudera Manager Admin Console:



> **Note:** Screenshots in this guide show version 5.9 of the Cloudera Manager Admin Console.

The backup and restore processes are configured, managed, and executed using replication schedules. Each replication schedule identifies a *source* and a *destination* for the given replication process. The replication process uses a *pull* model. When the replication process runs, the configured destination cluster accesses the given source cluster and transparently performs all tasks needed to copy the HDFS files to the destination cluster.

The destination cluster handles configuration and running the schedule. Typically, creating a backup replication schedule takes place on the backup cluster and creating a restore replication schedule takes place on the production cluster. Thus, as shown in this tutorial, the example production cluster, *cloudera-bdr-src-{1..4}.cloud.computers.com*, is the source for the backup replication schedule and the destination for the restore replication schedule.

Backing Up HDFS Files

The backup process begins at the Cloudera Manager Admin Console on the cluster designated as the backup, and includes these steps:

- Step 1: Establish a Peer Relationship to the Production Cluster on page 548
- Step 2: Configure the Replication Schedule for the Backup on page 549
- Step 3: Verify Successful Replication on page 550

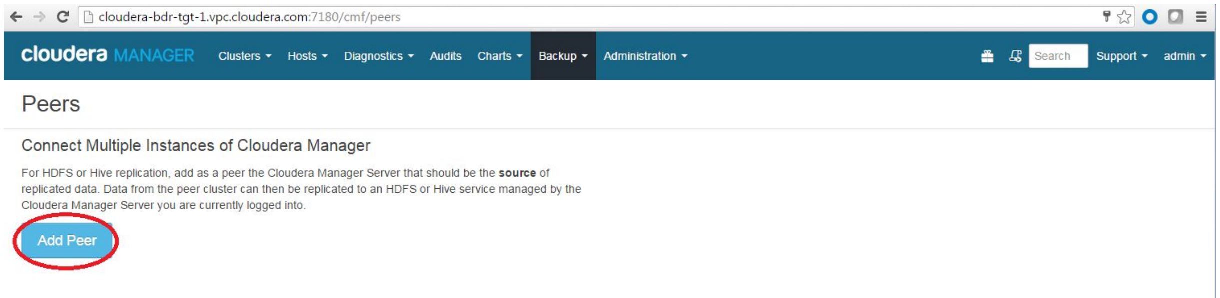
Step 1: Establish a Peer Relationship to the Production Cluster

For a backup, the destination is the backup cluster, and the source is the production cluster.

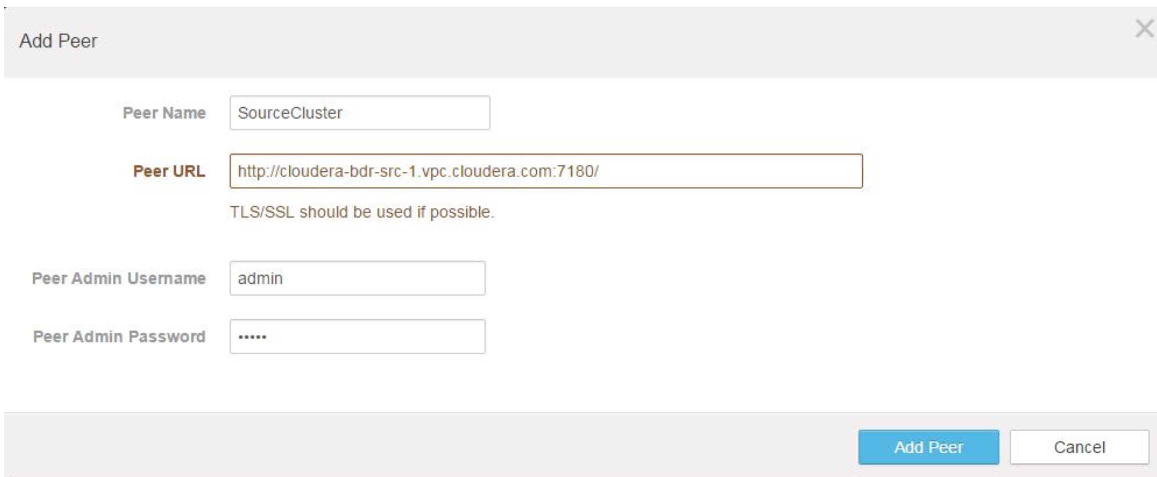
The cluster establishing the peer relationship gains access to the source cluster and can run the export command, list HDFS files, and read files for copying them to the destination cluster. These are all the actions performed by the replication process whenever the defined schedule goes into action.

Defining the replication starts from the Cloudera Manager Admin Console on the backup cluster.

1. Log in to Cloudera Manager Admin Console on the backup cluster.
2. Click the **Backup** tab and select **Peers** from the menu.
3. On the Peers page, click **Add Peer**:

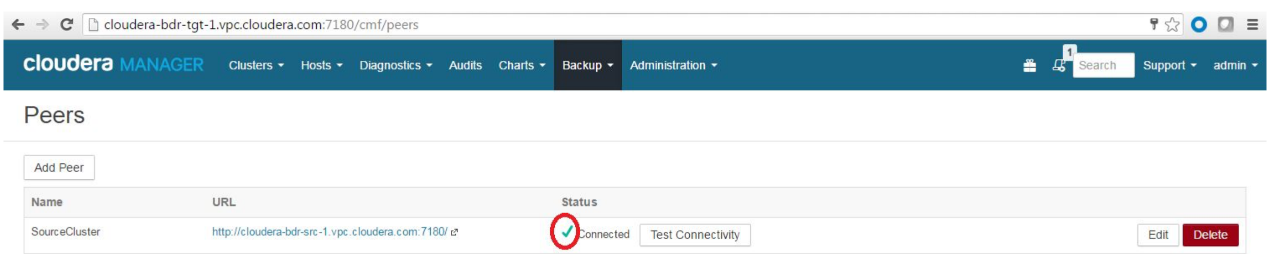


4. On the Add Peer page:
   a. **Peer Name** - Enter a meaningful name for the cluster that you want to back up. This peer name becomes available in the next step, to be selected as the source for the replication.
   b. **Peer URL** - Enter the URL for the Cloudera Manager Admin Console running on the master node of the cluster.
   c. **Peer Admin Username** - Enter the administrator user account for the production cluster.
   d. **Peer Admin Password** - Enter the password for the administrator account for the production cluster.



5. Click **Add Peer** to save your settings, connect to the production cluster, and establish this peer relationship.

After the system establishes and verifies the connection to the peer, the Peers page re-displays, showing the Status column as Connected (note the green check-mark):



With the peer relationship established from destination to source, create a schedule to replicate HDFS files from the source (production cluster) to the destination (backup cluster).

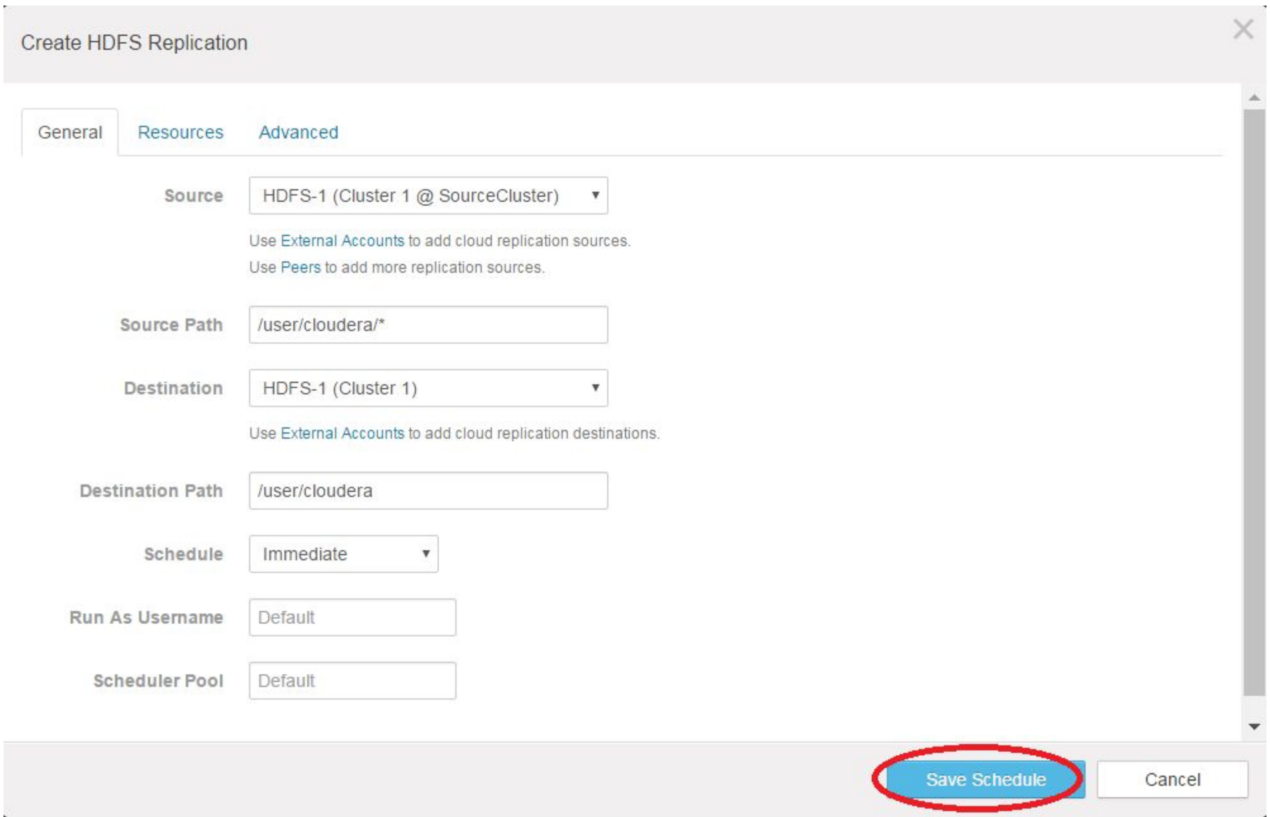
Step 2: Configure the Replication Schedule for the Backup

From the Cloudera Manager Admin Console on the backup cluster:

1. Click the **Backup** tab and select **Replication Schedules** from the menu.
2. On the Replication Schedules page, select **Create Schedule** > **HDFS Replication**.
3. On the Create HDFS Replication page, click the **General** tab to display the default schedule options:

a. **Source** - Make sure the cluster node selected in the drop-down is the production cluster (the cluster to be backed up).
b. **Source Path** - Specify the directory name on the production cluster holding the files to back up. Use an asterisk (*) on the directory name to specify that only the explicit directory and no others should be created on the destination. Without the asterisk, directories may be nested inside a containing directory on the destination.
c. **Destination** - Select the cluster to use as the target of the replication process, typically, the cluster to which you have logged in and the cluster to which you want to backup HDFS data.
d. **Destination Path** - Specify a directory name on the backup cluster.
e. **Schedule** - Immediate.
f. **Run As Username** - Leave as Default.
g. **Scheduler Pool** - Leave as Default.

4. Click **Save Schedule**.



The files are replicated from the source cluster to the backup cluster immediately.

When the process completes, the Replication Schedules page re-displays, showing a green check-mark and time-stamp in the Last Run column.

Step 3: Verify Successful Replication

Verify that the HDFS files are now on the backup cluster by using the HDFS File Browser or the `hadoop fs -ls` command (shown below):

```
hdfs@cloudera-bdr-tgt-1:~
[hdfs@cloudera-bdr-tgt-1 ~]$ hadoop fs -ls /user/cloudera
Found 9 items
-rw-r--r--   3 hdfs supergroup 16106127360 2016-09-01 14:17 /user/cloudera/large.0
-rw-r--r--   3 hdfs supergroup  5368709120 2016-09-01 14:29 /user/cloudera/medium.0
-rw-r--r--   3 hdfs supergroup  5368709120 2016-09-01 14:33 /user/cloudera/medium.1
-rw-r--r--   3 hdfs supergroup  5368709120 2016-09-01 14:35 /user/cloudera/medium.2
-rw-r--r--   3 hdfs supergroup  1073741824 2016-09-01 14:36 /user/cloudera/small.0
-rw-r--r--   3 hdfs supergroup  1073741824 2016-09-01 14:36 /user/cloudera/small.1
-rw-r--r--   3 hdfs supergroup  1073741824 2016-09-01 14:47 /user/cloudera/small.2
-rw-r--r--   3 hdfs supergroup  1073741824 2016-09-01 14:48 /user/cloudera/small.3
-rw-r--r--   3 hdfs supergroup  1073741824 2016-09-01 14:48 /user/cloudera/small.4
[hdfs@cloudera-bdr-tgt-1 ~]$
```

The HDFS files are now on both the production cluster and the backup cluster.

You can now use the backup as a source for a restore onto the production system when needed.

Configuring a Restore Replication Schedule

To restore data from a backup, configure a Restore Schedule on the cluster to which you want the data pulled. For this example, the replication schedule is created on the example production cluster and designed to pull HDFS files from the backup cluster to a different path on the production cluster, to enable testing the restore process in advance of any failure.

> **Note:** This approach lets you step through and test the process prior to using the schedule in a production environment.

Setting up a schedule for a restore follows the same pattern as setting up the backup, but with all actions initiated using the Cloudera Manager Admin Console on the production cluster.

To set up and test a replication schedule to restore HDFS from an existing backup copy, follow these steps:

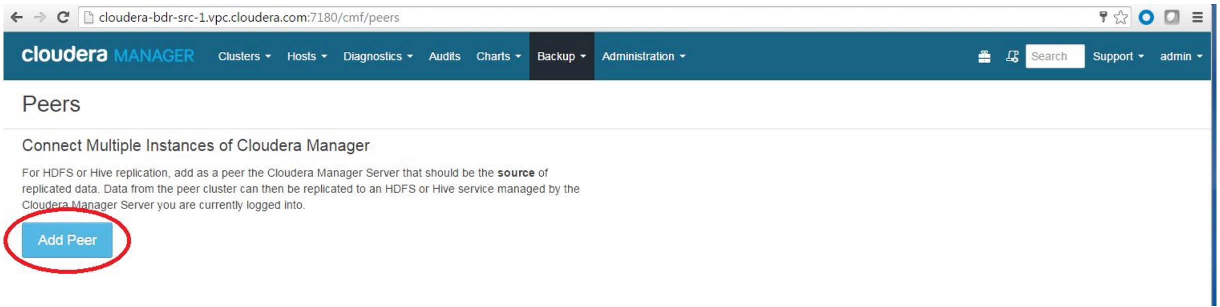
- Step 1: Establish a Peer Relationship to the Backup Cluster on page 551
- Step 2: Configure Replication Schedule to Test the Restore on page 552
- Step 3: Disable the Replication Schedule on page 553
- Step 4: Test the Restore Replication Schedule on page 554
- Step 5: Verify Successful Data Restoration on page 554

Step 1: Establish a Peer Relationship to the Backup Cluster

To restore HDFS files on the production cluster, establish a Peer relationship from the destination to the source for the restore.

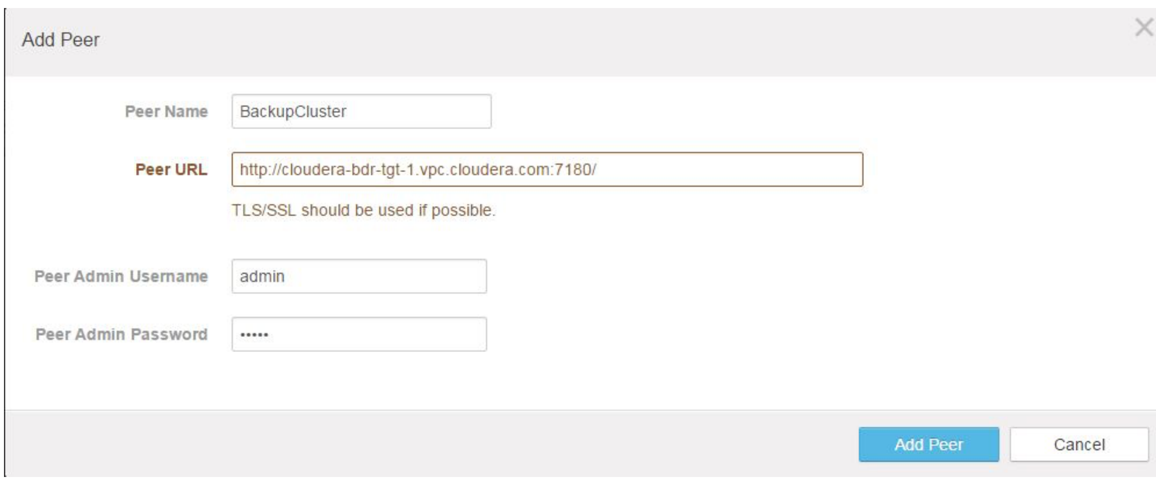
Log in to the Cloudera Manager Admin Console on the production cluster.

1. Click the Backup tab and select **Peers** from the menu.
2. On the Peers page, click **Add Peers**.



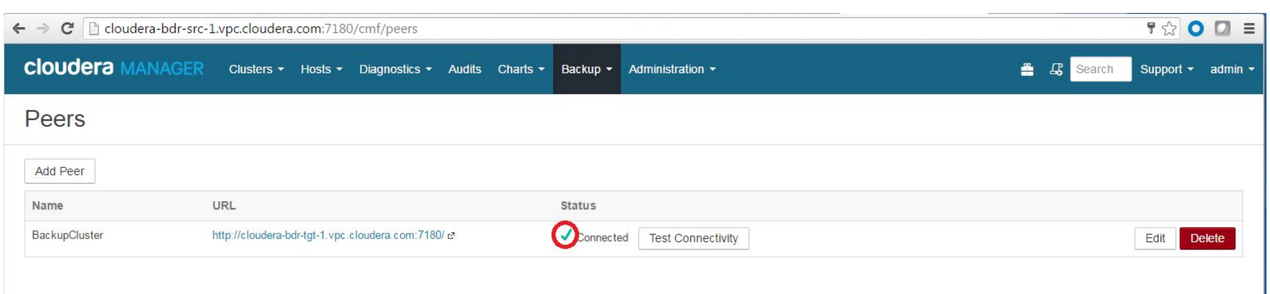
3. On the Add Peer page:

   a. **Peer Name** - Enter a meaningful name for the cluster that you want to restore data from, for example, *BackupCluster*.
   b. **Peer URL** - Enter the URL for the Cloudera Manager Admin Console running on the master node of the cluster.
   c. **Peer Admin Username** - Enter the administrator user name for the peer cluster.
   d. **Peer Admin Password** - Enter the password for the administrator user account.



4. Click **Add Peer** to save the settings, connect to the backup cluster, and establish the peer relationship.

Once the system connects and tests the peer relationship, the Peers page lists its name, URL, and Status (Connected):



The backup cluster is now available as a peer, for use in a Replication Schedule.

Step 2: Configure Replication Schedule to Test the Restore

The goal in these steps is to create a replication schedule that can be used when needed, in the future, but to leave it in a disabled state. However, because Replication Schedules cannot be created in a disabled state, you initially set the date far into the future and then disable the schedule in a subsequent step.

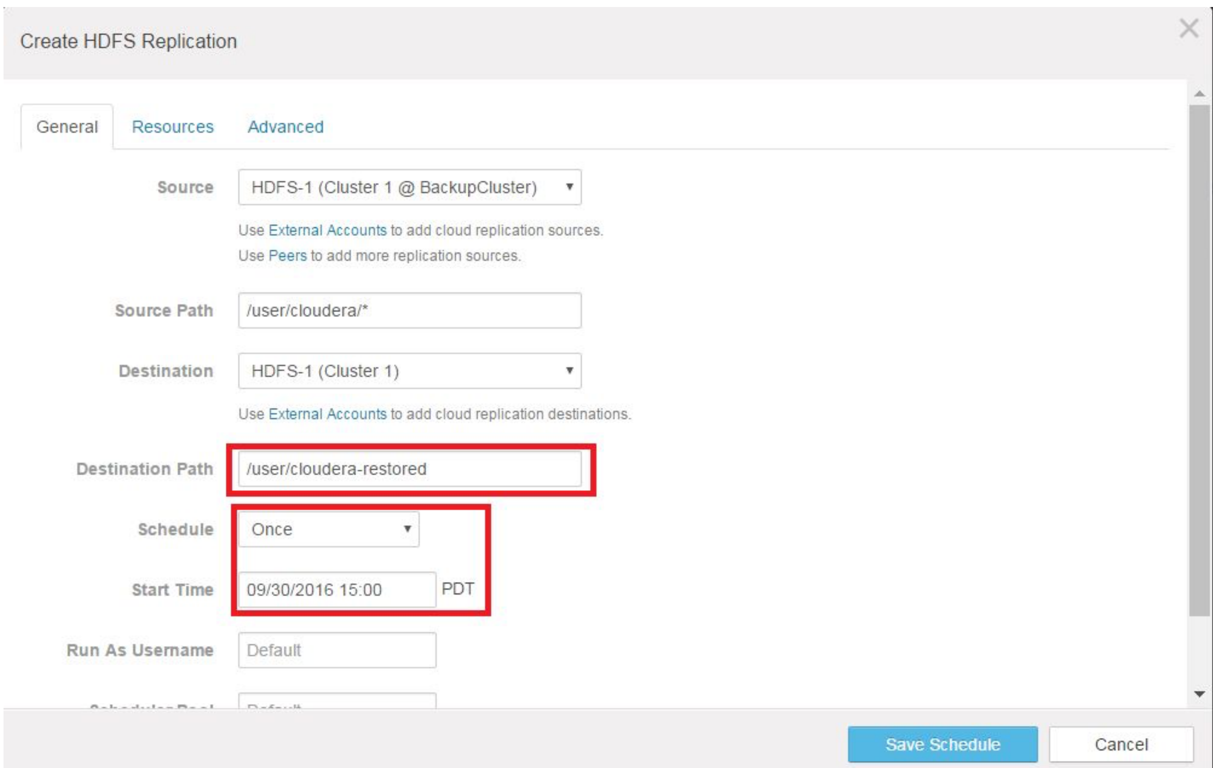
From the Cloudera Manager Admin Console on the production cluster:

1. Click the **Backup** tab and select **Replication Schedules** from the menu.

2. On the Replication Schedules page, select **Create Schedule** > **HDFS Replication**.
3. On the General settings tab of the Create HDFS Replication page:
   a. **Source** - Make sure the cluster node selected in the drop-down is the backup cluster.
   b. **Source Path** - Specify the directory name that you want to back up. Use an asterisk (*) on the directory name to specify that only the explicit directory and no others should be created on the destination.
   c. **Destination** - Select the cluster to use as the target for the replication process.
   d. **Destination Path** - Specify a directory name on the backup cluster.

   > **Important:** The path is a new directory on the example production cluster.

   e. **Schedule** - Set to a time far in the future, so that the schedule does not run as soon as it is saved.
   f. **Run As Username** - Leave as Default.
   g. **Scheduler Pool** - Leave as Default.



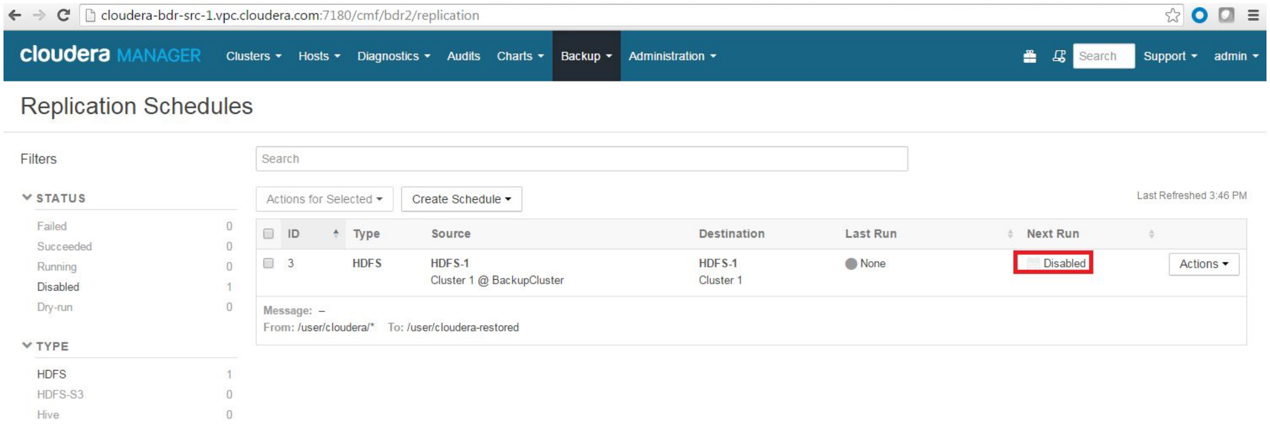
4. Click **Save Schedule**.

The Replication Schedule is saved and displays in the Replication Schedules list, with the future date listed in the Next Run column.

Before continuing, immediately disable this newly created Replication Schedule.

Step 3: Disable the Replication Schedule

1. On the Replication Schedules page, select the replication schedule.
2. From the **Actions** drop-down menu, select **Disable**. When the page refreshes, Disabled displays in the Next Run column:



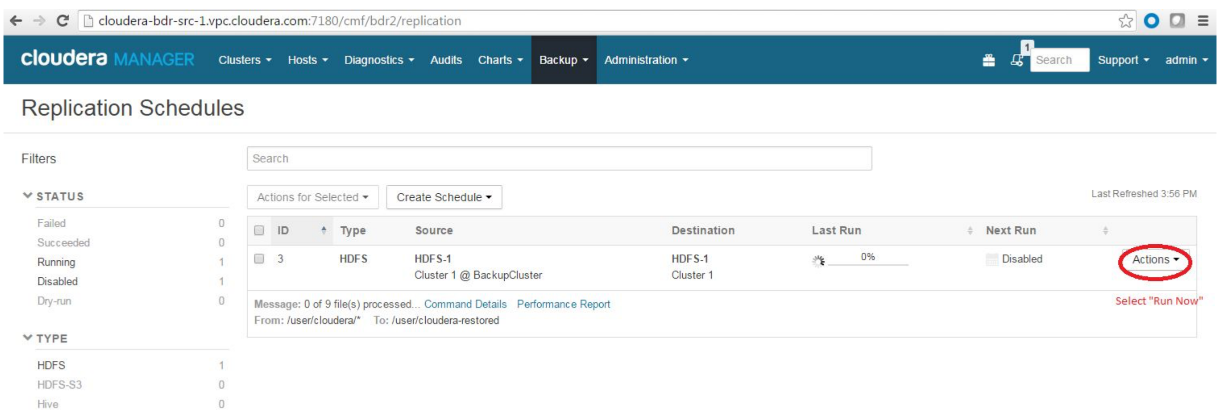
You can leave restore Replication Schedules pre-configured and disabled in this way so they are ready to use in the event of a catastrophic data loss. Before relying on this approach, test the schedule.

Step 4: Test the Restore Replication Schedule

The Replication Schedule defined in the example restores data to a specific directory path identified for the purpose of restoration (`/user/cloudera-restored`) rather than targeting the original source directory path.

From the Cloudera Manager Admin Console on the production cluster, with Replication Schedules page displayed:

1. On the Replication Schedules page, select the disabled replication schedule.
2. Select **Actions** > **Run Now**.



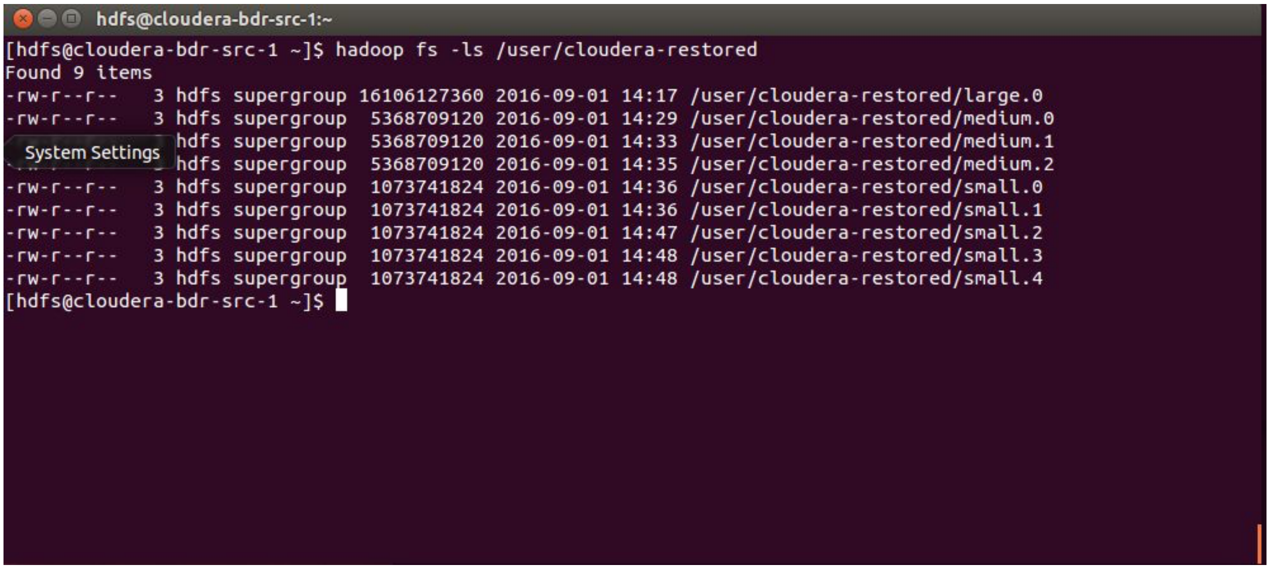
When the replication process completes, disable the Replication Schedule once again:

1. On the Replication Schedules page, select the replication schedule.
2. Select **Actions** > **Disable** .

You can now verify that the files have been replicated to the destination directory path.

Step 5: Verify Successful Data Restoration

To manually verify that your data has been restored to the source cluster, you can use the HDFS File Browser or the hadoop command-line, as shown here:



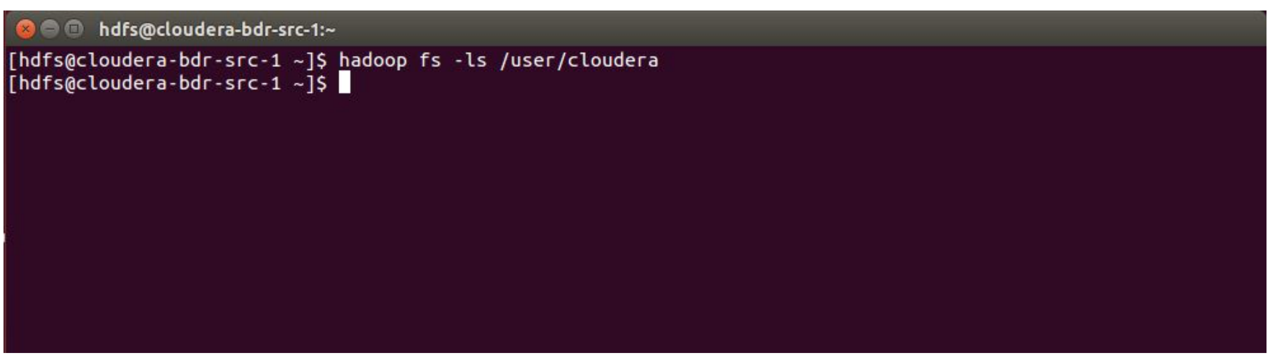
Compare the restored data in `/user/cloudera-restored` to the data in `/user/cloudera` to validate that the restore schedule operates as expected.

At this point, until you need to actually restore production HDFS data files, you can leave the Replication Schedule disabled.

> **Important:** In a production environment, only one Replication Schedule for a given dataset should be active at the same time. In this example, the Replication Schedule that created the backup has not been disabled yet, because the HDFS files from the backup cluster were restored to a different path on the example production cluster.

Recovering from Catastrophic Data Loss

With backup and restore Replication Schedules set up and validated, you can restore data when production data has been erroneously deleted, as shown in this screenshot:



In the event of an actual data loss on a production cluster, you should first disable any existing replication schedules for the affected datasets *before* activating a replication schedule for the restore, to avoid overwriting existing replicas on the backup cluster with defective files.

- Step 1: Disable the Backup Replication Schedule on page 556
- Step 2: Edit the Existing Replication Schedule on page 556
- Step 3: Run the Restore Replication Schedule on page 557
- Step 4: Return the Restore Replication Schedule to a Disabled State on page 557
- Step 5: Re-enable the Backup Replication Schedule on page 557

Step 1: Disable the Backup Replication Schedule

Disabling any existing replication schedule for HDFS backups can help prevent the replication of lost or corrupted data files over existing backups.

At the Cloudera Manager Admin Console on the backup cluster:

1. Select **Backup** > **Replication Schedules**.
2. On the Replication Schedules page, select the schedule.
3. From the Actions drop-down menu, select **Disable**:

When the Replication Schedules pages refreshes, you see **Disabled** in the **Next Run** column for the schedule.

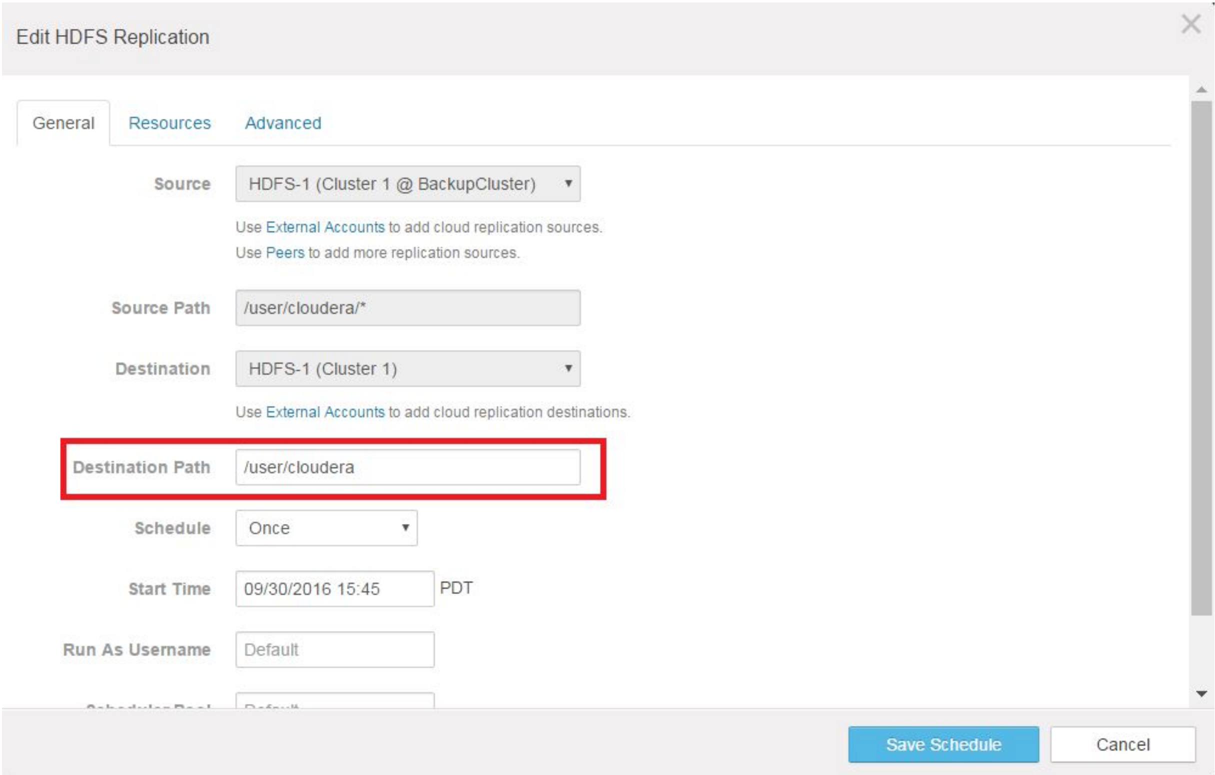
Step 2: Edit the Existing Replication Schedule

With the replication schedule disabled, you can edit the replication schedule verified previously (Step 4: Test the Restore Replication Schedule on page 554) and restore the data to the production cluster.

From the Cloudera Manager Admin Console on the production cluster:

1. Click the **Backup** tab and select **Replication Schedules** from the menu.
2. On the Replication Schedules page, select **Create Schedule** > **HDFS Replication**.
3. On the General settings tab of the Create HDFS Replication page:

    a. **Source** - Name of the backup cluster from which to pull the data. For this example, the source is the backup cluster.
    b. **Source Path** - The path on the backup cluster that contains the data you want to restore. Use the asterisk (*) at the end of the directory name to prevent extraneous sub-directories being created on the destination.
    c. **Destination** - The name of the cluster on which to restore the data, in which case, the example production cluster.
    d. **Destination Path** - Directory in which to restore the HDFS data files, in this case, the directory on the example production system.
    e. **Schedule** - Once.
    f. **Start Time** - Leave set to the future date and time that you originally defined in Step 2: Configure Replication Schedule to Test the Restore on page 552.
    g. **Run as Username** - Leave set as Default.



4. Click **Save Schedule**.

The settings for the Replication Schedule are saved, and the page refreshes. Because this replication schedule is currently disabled, you must actively run the schedule to restore the data.

Step 3: Run the Restore Replication Schedule

While still on the Replication Schedules page:

1. Select the edited replication schedule.
2. From the **Actions** drop-down menu, select **Run Now**.

The replication schedule executes and restores the HDFS files to the original location.

When the process completes, the Replication Schedules page displays the time-stamp and check-mark in the Last Run column for the schedule.

Step 4: Return the Restore Replication Schedule to a Disabled State

You can now disable the schedule, and after verifying that HDFS data has been successfully restored to the production cluster, you can re-enable the backup schedule. While still displaying the Replication Schedules page on the production cluster:

1. Select the replication schedule used for the restore.
2. Edit its configuration again, to point to a non-production directory.
3. Select **Actions** > **Disable**.

After confirming that the schedule has been disabled—you see **Disabled** in the Next Run column for this schedule—but before re-enabling the backup schedule, verify that the HDFS files have been restored to the production cluster.

Step 5: Re-enable the Backup Replication Schedule

With data restored to the production cluster and the replication schedule on the production cluster disabled, you can re-enable the replication schedule on the backup cluster.

1. At the backup cluster, log in to the Cloudera Manager Admin Console.
2. Select **Backup** > **Replication Schedules**.

3. On the Replication Schedules page, select the appropriate replication schedule to back up the production cluster.
4. From the **Actions** drop-down menu, select **Enable**.

This concludes the tutorial. In an actual production environment, you should configure replication schedules to regularly backup production systems. For restoring files from any backup, you can create and test a replication schedules in advance, as shown in this tutorial.

Alternatively, you can create a replication schedule to restore data specifically when needed. See How To Back Up and Restore Apache Hive Data Using Cloudera Enterprise BDR on page 534 for details.

See Backup and Disaster Recovery on page 473 and BDR Tutorials on page 534 for more information about Cloudera Enterprise BDR.

Cloudera Manager Administration

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 backup of the Cloudera Manager database is available.

To move the 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.

> **Important:**
>
> - The Cloudera Manager version on the destination host *must match* the version on the source host.
> - 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. 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 Datastores.
   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, username, and password.
6. Do the following on all cluster hosts:
   a. In `/etc/cloudera-scm-agent/config.ini`, update the `server_host` property to the new hostname.
   b. If you are replacing the Cloudera Manager database with a new database, and you are not using a backup of the original Cloudera Manager database, delete the `/var/lib/cloudera-scm-agent/cm_guid` file.
   c. Restart the agent using the following command:

```
$ sudo service cloudera-scm-agent restart
```

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.

Migrating from the Cloudera Manager Embedded PostgreSQL Database Server to an External PostgreSQL Database

Cloudera Manager provides an embedded PostgreSQL database server for demonstration and proof of concept deployments when creating a cluster. To remind users that this embedded database is not suitable for production, Cloudera Manager displays the banner text: "You are running Cloudera Manager in non-production mode, which uses an embedded PostgreSQL database. Switch to using a supported external database before moving into production."

If, however, you have already used the embedded database, and you are unable to redeploy a fresh cluster, then you *must* migrate to an external PostgreSQL database.

> **Note:** This procedure does *not* describe how to migrate to a database server other than PostgreSQL. Moving databases from one database server to a different type of database server is a complex process that requires modification of the schema and matching the data in the database tables to the new schema. It is strongly recommended that you engage with Cloudera Professional Services if you wish to perform a migration to an external database server other than PostgreSQL.

Prerequisites

Before migrating the Cloudera Manager embedded PostgreSQL database to an external PostgreSQL database, ensure that your setup meets the following conditions:

- The external PostgreSQL database server is running.
- The database server is configured to accept remote connections.
- The database server is configured to accept user logins using `md5`.
- No one has manually created any databases in the external database server for roles that will be migrated.

> **Note:** To view a list of databases in the external database server (requires default superuser permission):
>
> ```
> sudo -u postgres psql -l
> ```

- All health issues with your cluster have been resolved.

For details about configuring the database server, see Configuring and Starting the PostgreSQL Server.

> **Important:** Only perform the steps in Configuring and Starting the PostgreSQL Server. Do *not* proceed with the creation of databases as described in the subsequent section.

For large clusters, Cloudera recommends running your database server on a dedicated host. Engage Cloudera Professional Services or a certified database administrator to correctly tune your external database server.

Identify Roles that Use the Embedded Database Server

Before you can migrate to another database server, you must first identify the databases using the embedded database server. When the Cloudera Manager Embedded Database server is initialized, it creates the Cloudera Manager database and databases for roles in the Management Services. The Installation Wizard (which runs automatically the first time you log in to Cloudera Manager) or **Add Service** action for a cluster creates additional databases for roles when run. It is in this context that you identify which roles are used in the embedded database server.

To identify which roles are using the Cloudera Manager embedded database server:

1. Obtain and save the `cloudera-scm` superuser password from the embedded database server. You will need this password in subsequent steps:

```
head -1  /var/lib/cloudera-scm-server-db/data/generated_password.txt
```

2. Make a list of all services that are using the embedded database server. Then, after determining which services are not using the embedded database server, remove those services from the list. The `scm` database must remain in your list. Use the following table as a guide:

Table 36: Cloudera Manager Embedded Database Server Databases

Service	Role	Default Database Name	Default Username
Cloudera Manager Server		scm	scm
Cloudera Management Service	Activity Monitor	amon	amon
Hive	Hive Metastore Server	hive	hive
Hue	Hue Server	hue	7uu7uu7uhue

Service	Role	Default Database Name	Default Username
Cloudera Management Service	Navigator Audit Server	nav	nav
Cloudera Management Service	Navigator Metadata Server	navms	navms
Oozie	Oozie Server	oozie_oozie_server	oozie_oozie_server
Cloudera Management Service	Reports Manager	rman	rman
Sentry	Sentry Server	sentry	sentry

3. Verify which roles are using the embedded database. Roles using the embedded database server always use port 7432 (the default port for the embedded database) on the Cloudera Manager Server host.

   For Cloudera Management Services:

   a. Select **Cloudera Management Service** > **Configuration**, and type "7432" in the **Search** field.
   b. Confirm that the hostname for the services being used is the same hostname used by the Cloudera Manager Server.

   > **Note:**
   >
   > If any of the following fields contain the value "7432", then the service is using the embedded database:
   >
   > - **Activity Monitor**
   > - **Navigator Audit Server**
   > - **Navigator Metadata Server**
   > - **Reports Manager**

   For the Oozie Service:

   1. Select **Oozie** service > **Configuration**, and type "7432" in the **Search** field.
   2. Confirm that the hostname is the Cloudera Manager Server.

   For Hive, Hue, and Sentry Services:

   1. Select the specific service > **Configuration**, and type "database host" in the **Search** field.
   2. Confirm that the hostname is the Cloudera Manager Server.
   3. In the **Search** field, type "database port" and confirm that the port is 7432.
   4. Repeat these steps for each of the services (Hive, Hue and Sentry).

4. Verify the database names in the embedded database server match the database names on your list (Step 2). Databases that exist on the database server and not used by their roles do *not* need to be migrated. This step is to confirm that your list is correct.

   > **Note:** Do not add the `postgres`, `template0`, or `template1` databases to your list. These are used only by the PostgreSQL server.

```
psql -h localhost -p 7432 -U cloudera-scm -l
Password for user cloudera-scm: <password>

                                        List of databases
       Name         |        Owner        | Encoding |  Collate    |   Ctype    |
 Access
------------------+-------------------+----------+------------+------------+---------------
```

```
amon                 | amon                 | UTF8      | en_US.UTF8 | en_US.UTF8 |
hive                 | hive                 | UTF8      | en_US.UTF8 | en_US.UTF8 |
hue                  | hue                  | UTF8      | en_US.UTF8 | en_US.UTF8 |
nav                  | nav                  | UTF8      | en_US.UTF8 | en_US.UTF8 |
navms                | navms                | UTF8      | en_US.UTF8 | en_US.UTF8 |
oozie_oozie_server   | oozie_oozie_server   | UTF8      | en_US.UTF8 | en_US.UTF8 |
postgres             | cloudera-scm         | UTF8      | en_US.UTF8 | en_US.UTF8 |
rman                 | rman                 | UTF8      | en_US.UTF8 | en_US.UTF8 |
scm                  | scm                  | UTF8      | en_US.UTF8 | en_US.UTF8 |
sentry               | sentry               | UTF8      | en_US.UTF8 | en_US.UTF8 |
template0            | cloudera-scm         | UTF8      | en_US.UTF8 | en_US.UTF8 |
=c/"cloudera-scm"
template1            | cloudera-scm         | UTF8      | en_US.UTF8 | en_US.UTF8 |
=c/"cloudera-scm"
(12 rows)
```

You should now have a list of all roles and database names that use the embedded database server, and are ready to proceed with the migration of databases from the embedded database server to the external PostgreSQL database server.

Migrate Databases from the Embedded Database Server to the External PostgreSQL Database Server

While performing this procedure, ensure that the Cloudera Manager Agents remain running on all hosts. Unless otherwise specified, when prompted for a password use the `cloudera-scm` password.

> **Note:** After completing this migration, you cannot delete the `cloudera-scm postgres` superuser unless you remove the access privileges for the migrated databases. Minimally, you should change the `cloudera-scm postgres` superuser password.

1. In Cloudera Manager, stop the cluster services identified as using the embedded database server (see Identify Roles that Use the Embedded Database Server on page 561). Refer to Starting, Stopping, and Restarting Services on page 45 for details about how to stop cluster services. Be sure to stop the Cloudera Management Service as well. Also be sure to stop any services with dependencies on these services. The remaining CDH services will continue to run without downtime.

   > **Note:** If you do not stop the services from within Cloudera Manager before stopping Cloudera Manager Server from the command line, they will continue to run and maintain a network connection to the embedded database server. If this occurs, then the embedded database server will ignore any command line stop commands (Step 2) and require that you manually kill the process, which in turn causes the services to crash instead of stopping cleanly.

2. Navigate to **Hosts** > **All Hosts**, and make note of the number of roles assigned to hosts. Also take note whether or not they are in a commissioned state. You will need this information later to validate that your `scm` database was migrated correctly.
3. Stop the Cloudera Manager Server. To stop the server:

```
sudo service cloudera-scm-server stop
```

4. Obtain and save the embedded database superuser password (you will need this password in subsequent steps) from the `generated_password.txt` file:

```
head -1  /var/lib/cloudera-scm-server-db/data/generated_password.txt
```

5. Export the PostgreSQL user roles from the embedded database server to ensure the correct users, permissions, and passwords are preserved for database access. Passwords are exported as an `md5sum` and are not visible in plain text. To export the database user roles (you will need the `cloudera-scm` user password):

```
pg_dumpall -h localhost -p 7432 -U cloudera-scm -v --roles-only -f
"/var/tmp/cloudera_user_roles.sql"
```

6. Edit `/var/tmp/cloudera_user_roles.sql` to remove any `CREATE ROLE` and `ALTER ROLE` commands for databases not in your list. Leave the entries for `cloudera-scm` untouched, because this user role is used during the database import.
7. Export the data from each of the databases on your list you created in [Identify Roles that Use the Embedded Database Server](#) on page 561:

```
pg_dump -F c -h localhost -p 7432 -U cloudera-scm [database_name]  >
/var/tmp/[database_name]_db_backup-$(date +"%m-%d-%Y").dump
```

Following is a sample data export command for the `scm` database:

```
pg_dump -F c -h localhost -p 7432 -U cloudera-scm scm > /var/tmp/scm_db_backup-$(date
+%m-%d-%Y).dump
Password:
```

8. Stop and disable the embedded database server:

```
service cloudera-scm-server-db stop
chkconfig cloudera-scm-server-db off
```

Confirm that the embedded database server is stopped:

```
netstat -at | grep 7432
```

9. Back up the Cloudera Manager Server database configuration file:

```
cp /etc/cloudera-scm-server/db.properties /etc/cloudera-scm-server/db.properties.embedded
```

10. Copy the file `/var/tmp/cloudera_user_roles.sql` and the database dump files from the embedded database server host to `/var/tmp` on the external database server host:

```
cd /var/tmp
scp cloudera_user_roles.sql *.dump <user>@<postgres-server>:/var/tmp
```

11. Import the PostgreSQL user roles into the external database server.

The external PostgreSQL database server superuser password is required to import the user roles. If the superuser role has been changed, you will be prompted for the username and password.

> **Note:** Only run the command that applies to your context; do *not* execute both commands.

- To import users when using the default PostgreSQL superuser role:

```
sudo -u postgres psql -f /var/tmp/cloudera_user_roles.sql
```

- To import users when the superuser role has been changed:

```
psql -h <database-hostname> -p <database-port> -U <superuser> -f
/var/tmp/cloudera_user_roles.sql
```

For example:

```
psql -h pg-server.example.com -p 5432 -U postgres -f /var/tmp/cloudera_user_roles.sql
```

```
Password for user postgres
```

12. Import the Cloudera Manager database on the external server. First copy the database dump files from the Cloudera Manager Server host to your external PostgreSQL database server, and then import the database data:

    Note: To successfully run the `pg_restore` command, there must be an existing database on the database server to complete the connection; the existing database will *not* be modified. If the `-d <existing-database>` option is not included, then the `pg_restore` command will fail.

```
pg_restore -C -h <database-hostname> -p <database-port> -d <existing-database> -U
cloudera-scm -v <data-file>
```

    Repeat this import for each database.

    The following example is for the `scm` database:

```
pg_restore -C -h pg-server.example.com -p 5432 -d postgres -U cloudera-scm -v
/var/tmp/scm_server_db_backup-20180312.dump
pg_restore: connecting to database for restore
```

```
Password:
```

13. Update the Cloudera Manager Server database configuration file to use the external database server. Edit the `/etc/cloudera-scm-server/db.properties` file as follows:

    a. Update the `com.cloudera.cmf.db.host` value with the hostname and port number of the external database server.
    b. Change the `com.cloudera.cmf.db.setupType` value from "`EMBEDDED`" to "`EXTERNAL`".

14. Start the Cloudera Manager Server and confirm it is working:

```
service cloudera-scm-server start
```

    Note that if you start the Cloudera Manager GUI at this point, it may take up to five minutes after executing the start command before it becomes available.

    In Cloudera Manager Server, navigate to **Hosts** > **All Hosts** and confirm the number of roles assigned to hosts (this number should match what you found in Step 2); also confirm that they are in a commissioned state that matches what you observed in Step 2.

15. Update the role configurations to use the external database hostname and port number. Only perform this task for services where the database has been migrated.

    For Cloudera Management Services:

    1. Select **Cloudera Management Service** > **Configuration**, and type "7432" in the **Search** field.
    2. Change any database hostname properties from the embedded database to the external database hostname and port number.
    3. Click **Save Changes**.

    For the Oozie Service:

    a. Select **Oozie** service > **Configuration**, and type "7432" in the **Search** field.

   b. Change any database hostname properties from the embedded database to the external database hostname and port number.
   c. Click **Save Changes**.

   For Hive, Hue, and Sentry Services:

   1. Select the specific service > **Configuration**, and type "database host" in the **Search** field.
   2. Change the hostname from the embedded database name to the external database hostname.
   3. Click **Save Changes**.

16. Start the **Cloudera Management Service** and confirm that all management services are up and no health tests are failing.
17. Start all Services via the Cloudera Manager web UI. This should start all services that were stopped for the database migration. Confirm that all services are up and no health tests are failing.
18. On the embedded database server host, remove the embedded PostgreSQL database server:

   a. Make a backup of the `/var/lib/cloudera-scm-server-db/data` directory:

```
tar czvf /var/tmp/embedded_db_data_backup-$(date +”%m-%d-%Y”).tgz
/var/lib/cloudera-scm-server-db/data
```

   b. Remove the embedded database package:

```
For RHEL:
rpm --erase cloudera-manager-server-db-2

For Debian:
apt-get remove cloudera-manager-server-db-2
```

   c. Delete the `/var/lib/cloudera-scm-server-db/data` directory.

Managing the Cloudera Manager Server Log

Viewing the Log

To help you troubleshoot problems, you can view the Cloudera Manager Server log. You can view the logs in the Logs page or in specific pages for the log.

Viewing Cloudera Manager Server 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. Click ▸ to the left of Cloudera Manager and select the **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.

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
```

Cloudera Manager Agents

The Cloudera Manager Agent is a Cloudera Manager component that works with the Cloudera Manager Server to manage the processes that map to role instances.

In a Cloudera Manager managed cluster, you can only start or stop role instance processes using Cloudera Manager. Cloudera Manager uses an open source process management tool called `supervisord`, that starts processes, takes care of redirecting log files, notifying of process failure, setting the effective user ID of the calling process to the right user, and so on. Cloudera Manager supports automatically restarting a crashed process. It will also flag a role instance with a bad health flag if its process crashes repeatedly right after start up.

The Agent is started by `init.d` at start-up. It, in turn, contacts the Cloudera Manager Server and determines which processes should be running. The Agent is monitored as part of Cloudera Manager's host monitoring. If the Agent stops heartbeating, the host is marked as having bad health.

One of the Agent's main responsibilities is to start and stop processes. When the Agent detects a new process from the Server heartbeat, the Agent creates a directory for it in `/var/run/cloudera-scm-agent` and unpacks the configuration. It then contacts `supervisord`, which starts the process.

cm_processes

To enable Cloudera Manager to run scripts in subdirectories of `/var/run/cloudera-scm-agent`, (because `/var/run` is mounted `noexec` in many Linux distributions), Cloudera Manager mounts a `tmpfs`, named `cm_processes`, for process subdirectories.

A `tmpfs` defaults to a max size of 50% of physical RAM but this space is not allocated until its used, and tmpfs is paged out to swap if there is memory pressure.

The lifecycle actions of `cmprocesses` can be described by the following statements:

- Created when the Agent starts up for the first time with a new supervisord process.

- If it already exists without `noexec`, reused when the Agent is started using `start` and not recreated.
- Remounted if Agent is started using `clean_restart`.
- Unmounting and remounting cleans out the contents (since it is mounted as a tmpfs).
- Unmounted when the host is rebooted.
- Not unmounted when the Agent is stopped.

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**
  1. Stop all roles running on the host. See Starting, Stopping, and Restarting Role Instances on page 56.
  2. Run the following command:
     - 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**
  1. Stop all roles running on the host. See Starting, Stopping, and Restarting Role Instances on page 56.
  2. Run the following command:

- 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 are upgrading Cloudera Manager and the `supervisord` code has changed between your current version and the new one. To properly do this upgrade you need to restart supervisor too.
2. `supervisord` freezes 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.
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]`	`server_host, server_port, listening_port, listening_hostname, listening_ip`	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 559 and Ports Used by Cloudera Manager and Cloudera Navigator. The Cloudera Manager Agent configures its hostname automatically. You can also manually specify the hostname the Cloudera Manager Agent uses by updating the `listening_hostname` property. To manually specify the IP address the Cloudera Manager Agent uses, update the `listening_ip` property in the same file. To have a CNAME used throughout instead of the regular hostname, an Agent can be configured to use `listening_hostname=`*CNAME*. 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 `namenode.`*`mycluster`*`.`*`company`*`.com` instead of `machine1234.`*`mycluster`*`.`*`company`*`.com`. In this case, `namenode.`*`mycluster`* would be a CNAME for `machine1234.`*`mycluster`*, and the generated client configurations (and internal configurations as well) would use the CNAME.

Section	Property	Description
	`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`. **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. 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. 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.
	`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 • 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 64
[Cloudera]	mgmt_home	Directory to store Cloudera Management Service files. Default: /usr/share/cmf.
[JDBC]	cloudera_mysql_connector_jar, cloudera_oracle_connector_jar, cloudera_postgresql_jdbc_jar	Location of JDBC drivers. See Cloudera Manager and Managed Service Datastores. Default: • MySQL - /usr/share/java/mysql-connector-java.jar • Oracle - /usr/share/java/oracle-connector-java.jar • PostgreSQL - /usr/share/cmf/lib/postgresql-*version-build*.jdbc4.jar

Managing Cloudera Manager Agent Logs

Viewing Agent Logs

To help you troubleshoot problems, you can view the Cloudera Manager Agent logs. You can view the logs in the Logs page or in specific pages for the logs.

Viewing Cloudera Manager 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. Click ▸ to the left of Cloudera Manager and select the **Agent** checkbox.
5. Click **Search**.

For more information about the Logs page, see Logs.

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.

You can also view the Cloudera Manager Agent log at `/var/log/cloudera-scm-agent/cloudera-scm-agent.log` on the Agent hosts.

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.
> - If any user created scripts reference specific hostnames those must also be updated.
> - Due to the length and complexity of the following procedure, changing cluster hostnames is not recommended by Cloudera.

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 TLS/SSL certificates have been issued for any of the services and make sure to create new TLS/SSL certificates in advance for services protected by TLS/SSL. See Configuring 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/v14/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 the old hostname cluster principals.
         - If you are using an MIT KDC, remove old hostname cluster service principals from the KDC database using one of the following:
           – Use the `delprinc` command within `kadmin.local` interactive shell.

             OR

           – 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 noncluster service principal entries. Make sure that the default `krbtgt` and other principals you created, or that 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`.

         - For an Active Directory KDC, an AD administrator must manually delete the principals for the old hostname from Active Directory.
      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: `rmr /hadoop-ha/nameservice1`.
   d. If you *are not* running JobTracker in a high availability configuration, delete the HA znode: `rmr /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.

Alerts

An ***alert*** is an event that is considered especially noteworthy and is triggered by a selected event. Alerts are shown with an Alert badge when they appear in a list of events. You can configure the Alert Publisher to send alert notifications by email or by SNMP trap to a trap receiver.

Service instances of type HDFS, MapReduce, and HBase (and their associated roles) can generate alerts if so configured. Alerts can also be configured for the monitoring roles that are a part of the Cloudera Management Service.

The settings to enable or disable specific alerts are found under the Configuration tab for the services to which they pertain. See Configuring Alerts and for more information on setting up alerting.

For information about configuring the Alert Publisher to send email or SNMP notifications for alerts, see Configuring Alert Delivery.

Viewing What Alerts are Enabled and Disabled

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

Do one of the following:

- Select **Administration** > **Alerts**.
- Display the All Alerts Summary page:

  1. Do one of the following:

     - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
     - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

  2. Click the **Instances** tab.
  3. Click an **Alert Publisher** role.
  4. Click the **All Alerts Summary** tab.

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 **Edit** 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 and you can also configure a custom script that runs in response to an alert.

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. Select the **Administration** > **Alerts** tab and click **Edit** to the right of **Recipient(s)**.
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 Using Cloudera Manager](#) on page 13.

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 Using Cloudera Manager on page 13.
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 583.

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 **Home** > **Status** tab, 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 Using Cloudera Manager on page 13.

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 **Home** > **Status** tab, 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 **SMNP Mib** link.

Configuring Custom Alert Scripts

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 583.

You can configure the Alert Publisher to run a user-written script in response to an alert. The Alert Publisher passes a single argument to the script that is a UTF-8 JSON file containing a list of alerts. The script runs on the host where the Alert Publisher service is running and must have read and execute permissions for the **cloudera-scm** user. Only one instance of a script runs at a time. The standard out and standard error messages from the script are logged to the Alert Publisher log file.

You use the **Alert Publisher: Maximum Batch Size** and **Alert Publisher: Maximum Batch interval** to configure when the Alert Publisher delivers alerts. See Configuring Alerts.

To configure the Alert Publisher to deliver alerts using a script:

1. Save the script on the host where the Alert Publisher role is running.
2. Change the owner of the file to `cloudera-scm` and set its permissions to read and execute:

```
$ sudo chown cloudera-scm:cloudera-scm path_to_script
$ sudo chmod u+rx path_to_script
```

3. Open the Cloudera Manager Admin console and select **Clusters** > **Cloudera Management Service**.
4. Click the **Configuration** tab.
5. Select **Scope** > **Alert Publisher** .
6. Enter the path to the script in the **Custom Alert Script** property.
7. Click **Save Changes** to commit the changes.

Sample JSON Alert File

When a custom script runs, it passes a JSON file that contains the alerts. For example:


```
[ {
  "body" : {
    "alert" : {
      "content" : "The health test result for MAPREDUCE_HA_JOB_TRACKER_HEALTH has become
 bad: JobTracker summary: myCluster.com (Availability: Active, Health: Bad). This health
 test reflects the health of the active JobTracker.",
      "timestamp" : {
        "iso8601" : "2015-06-11T03:52:56Z",
        "epochMs" : 1433994776083
      },
      "source" :
"http://myCluster.com:7180/cmf/eventRedirect/89521139-0859-4bef-bf65-eb141e63dbba",
      "attributes" : {
        "__persist_timestamp" : [ "1433994776172" ],
        "ALERT_SUPPRESSED" : [ "false" ],
        "HEALTH_TEST_NAME" : [ "MAPREDUCE_HA_JOB_TRACKER_HEALTH" ],
        "SEVERITY" : [ "CRITICAL" ],
        "HEALTH_TEST_RESULTS" : [ {
          "content" : "The health test result for MAPREDUCE_HA_JOB_TRACKER_HEALTH has
become bad: JobTracker summary: myCluster.com (Availability: Active, Health: Bad). This
 health test reflects the health of the active JobTracker.",
          "testName" : "MAPREDUCE_HA_JOB_TRACKER_HEALTH",
          "eventCode" : "EV_SERVICE_HEALTH_CHECK_BAD",
          "severity" : "CRITICAL"
        } ],
        "CLUSTER_DISPLAY_NAME" : [ "Cluster 1" ],
        "ALERT" : [ "true" ],
        "CATEGORY" : [ "HEALTH_CHECK" ],
        "BAD_TEST_RESULTS" : [ "1" ],
        "SERVICE_TYPE" : [ "MAPREDUCE" ],
        "EVENTCODE" : [ "EV_SERVICE_HEALTH_CHECK_BAD", "EV_SERVICE_HEALTH_CHECK_GOOD"
],
        "ALERT_SUMMARY" : [ "The health of service MAPREDUCE-1 has become bad." ],
        "CLUSTER_ID" : [ "1" ],
        "SERVICE" : [ "MAPREDUCE-1" ],
        "__uuid" : [ "89521139-0859-4bef-bf65-eb141e63dbba" ],
        "CLUSTER" : [ "Cluster 1" ],
       "CURRENT_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test result
 for MAPREDUCE_HA_JOB_TRACKER_HEALTH has become bad: JobTracker summary: myCluster.com
 (Availability: Active, Health: Bad). This health test reflects the health of the active

JobTracker.\",\"testName\":\"MAPREDUCE_HA_JOB_TRACKER_HEALTH\",\"eventCode\":\"EV_SERVICE_HEALTH_CHECK_BAD\",\"severity\":\"CRITICAL\"}",
 "{\"content\":\"The health test result for MAPREDUCE_TASK_TRACKERS_HEALTHY has become
 good: Healthy TaskTracker: 3. Concerning TaskTracker: 0. Total TaskTracker: 3. Percent
 healthy: 100.00%. Percent healthy or concerning:
100.00%.\",\"testName\":\"MAPREDUCE_TASK_TRACKERS_HEALTHY\",\"eventCode\":\"EV_SERVICE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "PREVIOUS_HEALTH_SUMMARY" : [ "GREEN" ],
        "CURRENT_HEALTH_SUMMARY" : [ "RED" ],
        "MONITOR_STARTUP" : [ "false" ],
        "PREVIOUS_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test
result for MAPREDUCE_HA_JOB_TRACKER_HEALTH has become good: JobTracker summary:
myCluster.com (Availability: Active, Health:
Good)\",\"testName\":\"MAPREDUCE_HA_JOB_TRACKER_HEALTH\",\"eventCode\":\"EV_SERVICE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for MAPREDUCE_TASK_TRACKERS_HEALTHY has become
 good: Healthy TaskTracker: 3. Concerning TaskTracker: 0. Total TaskTracker: 3. Percent
 healthy: 100.00%. Percent healthy or concerning:
100.00%.\",\"testName\":\"MAPREDUCE_TASK_TRACKERS_HEALTHY\",\"eventCode\":\"EV_SERVICE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "SERVICE_DISPLAY_NAME" : [ "MAPREDUCE-1" ]
      }
    }
  },
  "header" : {
    "type" : "alert",
    "version" : 2
  }
}, {
  "body" : {
    "alert" : {
```

```
      "content" : "The health test result for JOB_TRACKER_SCM_HEALTH has become bad: 
This role's process exited. This role is supposed to be started.",
      "timestamp" : {
        "iso8601" : "2015-06-11T03:52:56Z",
        "epochMs" : 1433994776083
      },
      "source" : 
"http://myCluster.com:7180/cmf/eventRedirect/67b4d1c4-791b-428e-a9ea-8a09d4885f5d",
      "attributes" : {
        "__persist_timestamp" : [ "1433994776173" ],
        "ALERT_SUPPRESSED" : [ "false" ],
        "HEALTH_TEST_NAME" : [ "JOB_TRACKER_SCM_HEALTH" ],
        "SEVERITY" : [ "CRITICAL" ],
        "ROLE" : [ "MAPREDUCE-1-JOBTRACKER-10624c438dee9f17211d3f33fa899957" ],
        "HEALTH_TEST_RESULTS" : [ {
         "content" : "The health test result for JOB_TRACKER_SCM_HEALTH has become bad:
 This role's process exited. This role is supposed to be started.",
          "testName" : "JOB_TRACKER_SCM_HEALTH",
          "eventCode" : "EV_ROLE_HEALTH_CHECK_BAD",
          "severity" : "CRITICAL"
        } ],
        "CLUSTER_DISPLAY_NAME" : [ "Cluster 1" ],
        "HOST_IDS" : [ "75e763c2-8d22-47a1-8c80-501751ae0db7" ],
        "ALERT" : [ "true" ],
        "ROLE_TYPE" : [ "JOBTRACKER" ],
        "CATEGORY" : [ "HEALTH_CHECK" ],
        "BAD_TEST_RESULTS" : [ "1" ],
        "SERVICE_TYPE" : [ "MAPREDUCE" ],
        "EVENTCODE" : [ "EV_ROLE_HEALTH_CHECK_BAD", "EV_ROLE_HEALTH_CHECK_GOOD", 
"EV_ROLE_HEALTH_CHECK_DISABLED" ],
        "ALERT_SUMMARY" : [ "The health of role jobtracker (nightly-1) has become bad."
 ],
        "CLUSTER_ID" : [ "1" ],
        "SERVICE" : [ "MAPREDUCE-1" ],
        "__uuid" : [ "67b4d1c4-791b-428e-a9ea-8a09d4885f5d" ],
        "CLUSTER" : [ "Cluster 1" ],
       "CURRENT_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test result
 for JOB_TRACKER_SCM_HEALTH has become bad: This role's process exited. This role is 
supposed to be 
started.\",\"testName\":\"JOB_TRACKER_SCM_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_BAD\",\"severity\":\"CRITICAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become 
good: This role encountered 0 unexpected exit(s) in the previous 5 
minute(s).\",\"testName\":\"JOB_TRACKER_UNEXPECTED_EXITS\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_FILE_DESCRIPTOR has become good:
 Open file descriptors: 244. File descriptor limit: 32,768. Percentage in use: 
0.74%.\",\"testName\":\"JOB_TRACKER_FILE_DESCRIPTOR\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_SWAP_MEMORY_USAGE has become 
good: 0 B of swap memory is being used by this role's 
process.\",\"testName\":\"JOB_TRACKER_SWAP_MEMORY_USAGE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE has 
become good: This role's Log Directory (/var/log/hadoop-0.20-mapreduce) is on a filesystem
 with more than 20.00% of its space 
free.\",\"testName\":\"JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HOST_HEALTH has become good: 
The health of this role's host is 
good.\",\"testName\":\"JOB_TRACKER_HOST_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_WEB_METRIC_COLLECTION has become
 good: The web server of this role is responding with metrics. The most recent collection
 took 49 
millisecond(s).\",\"testName\":\"JOB_TRACKER_WEB_METRIC_COLLECTION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_GC_DURATION has become good: 
Average time spent in garbage collection was 0 second(s) (0.00%) per minute over the 
previous 5 
minute(s).\",\"testName\":\"JOB_TRACKER_GC_DURATION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE
 has become disabled: Test disabled because role is not configured to dump heap when 
out of memory. Test of whether this role's heap dump directory has enough free 
space.\",\"testName\":\"JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_DISABLED\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "CURRENT_HEALTH_SUMMARY" : [ "RED" ],
        "PREVIOUS_HEALTH_SUMMARY" : [ "GREEN" ],
        "MONITOR_STARTUP" : [ "false" ],
        "ROLE_DISPLAY_NAME" : [ "jobtracker (nightly-1)" ],
```

```
        "PREVIOUS_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test
result for JOB_TRACKER_SCM_HEALTH has become good: This role's status is as expected.
The role is
started.\",\"testName\":\"JOB_TRACKER_SCM_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become
good: This role encountered 0 unexpected exit(s) in the previous 5
minute(s).\",\"testName\":\"JOB_TRACKER_UNEXPECTED_EXITS\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_FILE_DESCRIPTOR has become good:
 Open file descriptors: 244. File descriptor limit: 32,768. Percentage in use:
0.74%.\",\"testName\":\"JOB_TRACKER_FILE_DESCRIPTOR\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_SWAP_MEMORY_USAGE has become
good: 0 B of swap memory is being used by this role's
process.\",\"testName\":\"JOB_TRACKER_SWAP_MEMORY_USAGE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE has
become good: This role's Log Directory (/var/log/hadoop-0.20-mapreduce) is on a filesystem
 with more than 20.00% of its space
free.\",\"testName\":\"JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HOST_HEALTH has become good:
The health of this role's host is
good.\",\"testName\":\"JOB_TRACKER_HOST_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_WEB_METRIC_COLLECTION has become
 good: The web server of this role is responding with metrics. The most recent collection
 took 49
millisecond(s).\",\"testName\":\"JOB_TRACKER_WEB_METRIC_COLLECTION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_GC_DURATION has become good:
Average time spent in garbage collection was 0 second(s) (0.00%) per minute over the
previous 5
minute(s).\",\"testName\":\"JOB_TRACKER_GC_DURATION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE
 has become disabled: Test disabled because role is not configured to dump heap when
out of memory. Test of whether this role's heap dump directory has enough free
space.\",\"testName\":\"JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_DISABLED\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "SERVICE_DISPLAY_NAME" : [ "MAPREDUCE-1" ],
        "HOSTS" : [ "myCluster.com" ]
      }
    }
  },
  "header" : {
    "type" : "alert",
    "version" : 2
  }
}, {
  "body" : {
    "alert" : {
      "content" : "The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become
bad: This role encountered 1 unexpected exit(s) in the previous 5 minute(s).This included
 1 exit(s) due to OutOfMemory errors. Critical threshold: any.",
      "timestamp" : {
        "iso8601" : "2015-06-11T03:53:41Z",
        "epochMs" : 1433994821940
      },
      "source" :
"http://myCluster.com:7180/cmf/eventRedirect/b8c4468d-08c2-4b5b-9bda-2bef892ba3f5",
      "attributes" : {
        "__persist_timestamp" : [ "1433994822027" ],
        "ALERT_SUPPRESSED" : [ "false" ],
        "HEALTH_TEST_NAME" : [ "JOB_TRACKER_UNEXPECTED_EXITS" ],
        "SEVERITY" : [ "CRITICAL" ],
        "ROLE" : [ "MAPREDUCE-1-JOBTRACKER-10624c438dee9f17211d3f33fa899957" ],
        "HEALTH_TEST_RESULTS" : [ {
         "content" : "The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become
 bad: This role encountered 1 unexpected exit(s) in the previous 5 minute(s).This included
 1 exit(s) due to OutOfMemory errors. Critical threshold: any.",
          "testName" : "JOB_TRACKER_UNEXPECTED_EXITS",
          "eventCode" : "EV_ROLE_HEALTH_CHECK_BAD",
          "severity" : "CRITICAL"
        } ],
        "CLUSTER_DISPLAY_NAME" : [ "Cluster 1" ],
        "HOST_IDS" : [ "75e763c2-8d22-47a1-8c80-501751ae0db7" ],
        "ALERT" : [ "true" ],
        "ROLE_TYPE" : [ "JOBTRACKER" ],
        "CATEGORY" : [ "HEALTH_CHECK" ],
```

```
        "BAD_TEST_RESULTS" : [ "1" ],
        "SERVICE_TYPE" : [ "MAPREDUCE" ],
        "EVENTCODE" : [ "EV_ROLE_HEALTH_CHECK_BAD", "EV_ROLE_HEALTH_CHECK_GOOD",
"EV_ROLE_HEALTH_CHECK_DISABLED" ],
        "ALERT_SUMMARY" : [ "The health of role jobtracker (nightly-1) has become bad."
 ],
        "CLUSTER_ID" : [ "1" ],
        "SERVICE" : [ "MAPREDUCE-1" ],
        "__uuid" : [ "b8c4468d-08c2-4b5b-9bda-2bef892ba3f5" ],
        "CLUSTER" : [ "Cluster 1" ],
       "CURRENT_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test result
 for JOB_TRACKER_SCM_HEALTH has become bad: This role's process exited. This role is
supposed to be
started.\",\"testName\":\"JOB_TRACKER_SCM_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_BAD\",\"severity\":\"CRITICAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become bad:
 This role encountered 1 unexpected exit(s) in the previous 5 minute(s).This included
1 exit(s) due to OutOfMemory errors. Critical threshold:
any.\",\"testName\":\"JOB_TRACKER_UNEXPECTED_EXITS\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_BAD\",\"severity\":\"CRITICAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_FILE_DESCRIPTOR has become good:
 Open file descriptors: 244. File descriptor limit: 32,768. Percentage in use:
0.74%.\",\"testName\":\"JOB_TRACKER_FILE_DESCRIPTOR\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_SWAP_MEMORY_USAGE has become
good: 0 B of swap memory is being used by this role's
process.\",\"testName\":\"JOB_TRACKER_SWAP_MEMORY_USAGE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE has
become good: This role's Log Directory (/var/log/hadoop-0.20-mapreduce) is on a filesystem
 with more than 20.00% of its space
free.\",\"testName\":\"JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HOST_HEALTH has become good:
The health of this role's host is
good.\",\"testName\":\"JOB_TRACKER_HOST_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_WEB_METRIC_COLLECTION has become
 good: The web server of this role is responding with metrics. The most recent collection
 took 49
millisecond(s).\",\"testName\":\"JOB_TRACKER_WEB_METRIC_COLLECTION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_GC_DURATION has become good:
Average time spent in garbage collection was 0 second(s) (0.00%) per minute over the
previous 5
minute(s).\",\"testName\":\"JOB_TRACKER_GC_DURATION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE
 has become disabled: Test disabled because role is not configured to dump heap when
out of memory. Test of whether this role's heap dump directory has enough free
space.\",\"testName\":\"JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_DISABLED\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "CURRENT_HEALTH_SUMMARY" : [ "RED" ],
        "PREVIOUS_HEALTH_SUMMARY" : [ "RED" ],
        "MONITOR_STARTUP" : [ "false" ],
        "ROLE_DISPLAY_NAME" : [ "jobtracker (nightly-1)" ],
        "PREVIOUS_COMPLETE_HEALTH_TEST_RESULTS" : [ "{\"content\":\"The health test
result for JOB_TRACKER_SCM_HEALTH has become bad: This role's process exited. This role
 is supposed to be
started.\",\"testName\":\"JOB_TRACKER_SCM_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_BAD\",\"severity\":\"CRITICAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_UNEXPECTED_EXITS has become
good: This role encountered 0 unexpected exit(s) in the previous 5
minute(s).\",\"testName\":\"JOB_TRACKER_UNEXPECTED_EXITS\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_FILE_DESCRIPTOR has become good:
 Open file descriptors: 244. File descriptor limit: 32,768. Percentage in use:
0.74%.\",\"testName\":\"JOB_TRACKER_FILE_DESCRIPTOR\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_SWAP_MEMORY_USAGE has become
good: 0 B of swap memory is being used by this role's
process.\",\"testName\":\"JOB_TRACKER_SWAP_MEMORY_USAGE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE has
become good: This role's Log Directory (/var/log/hadoop-0.20-mapreduce) is on a filesystem
 with more than 20.00% of its space
free.\",\"testName\":\"JOB_TRACKER_LOG_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HOST_HEALTH has become good:
The health of this role's host is
good.\",\"testName\":\"JOB_TRACKER_HOST_HEALTH\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_WEB_METRIC_COLLECTION has become
 good: The web server of this role is responding with metrics. The most recent collection
 took 49
millisecond(s).\",\"testName\":\"JOB_TRACKER_WEB_METRIC_COLLECTION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_GC_DURATION has become good:
```

```
Average time spent in garbage collection was 0 second(s) (0.00%) per minute over the
previous 5
minute(s).\",\"testName\":\"JOB_TRACKER_GC_DURATION\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_GOOD\",\"severity\":\"INFORMATIONAL\"}",
 "{\"content\":\"The health test result for JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE
 has become disabled: Test disabled because role is not configured to dump heap when
out of memory. Test of whether this role's heap dump directory has enough free
space.\",\"testName\":\"JOB_TRACKER_HEAP_DUMP_DIRECTORY_FREE_SPACE\",\"eventCode\":\"EV_ROLE_HEALTH_CHECK_DISABLED\",\"severity\":\"INFORMATIONAL\"}"
 ],
        "SERVICE_DISPLAY_NAME" : [ "MAPREDUCE-1" ],
        "HOSTS" : [ "myCluster.com" ]
      }
    }
  },
  "header" : {
    "type" : "alert",
    "version" : 2
  }
} ]
```


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 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 and custom alert scripts
- Backup and disaster recovery
- Operational reports
- Cloudera Navigator
- Commands such as Rolling Restart, History and Rollback, and Send Diagnostic Data
- Cluster Utilization Reports on page 351.

For details see What's the Difference Between Cloudera Express and Cloudera Enterprise?.

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.

Cloudera offers the following two types of licenses:

- **Cloudera Express**

  A free and unlimited license that provides access to CDH, Cloudera's Apache Hadoop distribution and a subset of cluster management features available with Cloudera Manager.

- **Cloudera Enterprise**

Cloudera Enterprise is available on a subscription basis in five editions, each designed around how you use the platform:

- **Basic Edition** provides superior support and advanced management for core Apache Hadoop.
- **Data Engineering Edition** for programmatic data preparation and predictive modeling.
- **Operational Database Edition** for online applications with real-time serving needs.
- **Analytic Database Edition** for BI and SQL analytics.
- **Enterprise Data Hub Edition** provides for complete use of the platform.

All editions are available in your environment of choice: cloud, on-premise, or a hybrid deployment. For more information, see the Cloudera Enterprise Data Sheet.

License Expiration

When a Cloudera Enterprise license expires, the following occurs:

- Cloudera Enterprise 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 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 Enterprise Data Hub Edition Trial

To start a trial, on the License page, click **Try Cloudera Enterprise Enterprise Data Hub Edition for 60 Days**.

1. Cloudera Manager displays a pop-up describing the features enabled with Cloudera Enterprise 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:

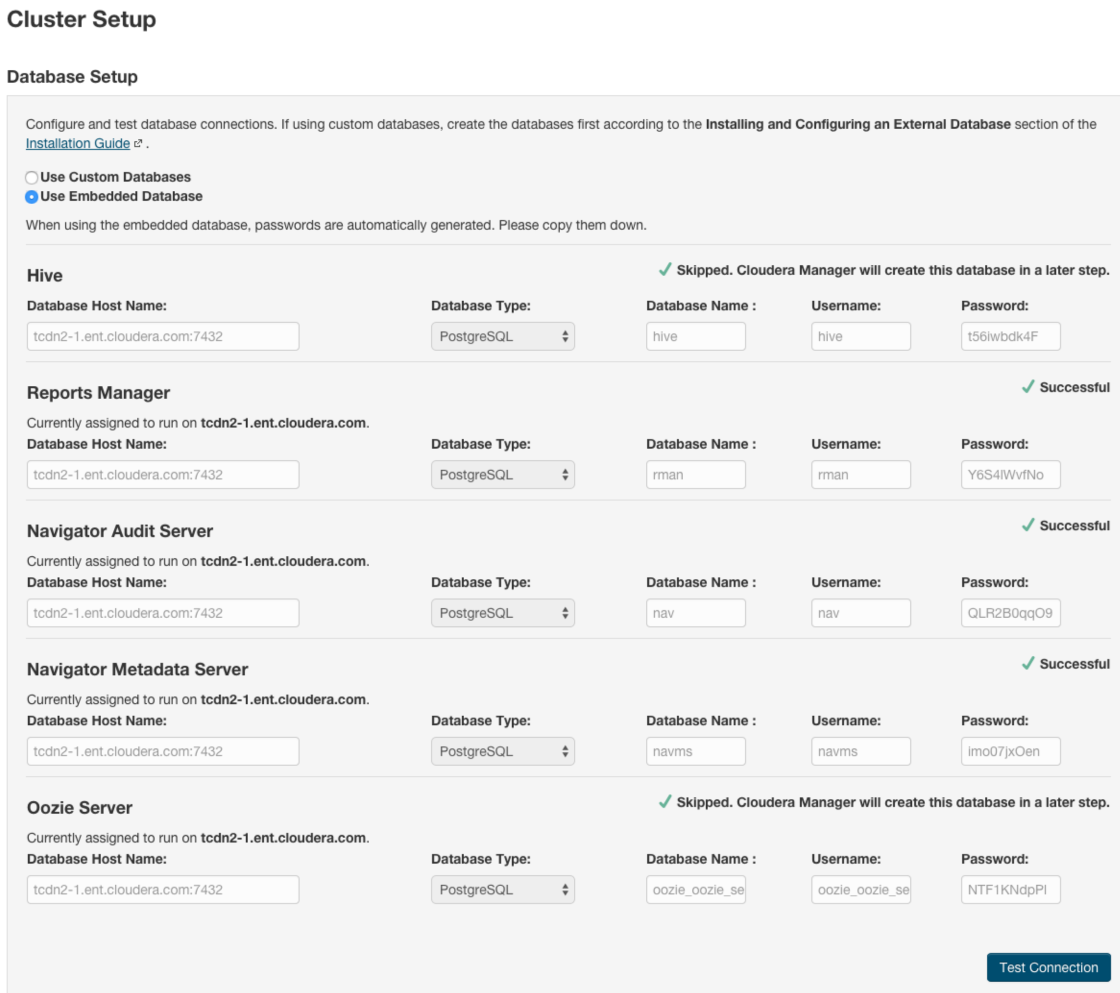
   - Range of hostnames (without the domain portion)

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. 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.



- Select **Use Custom Databases** to specify external database host, enter the 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 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 239.

6. 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 **Cluster Setup Review Changes** screen displays.

7. 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.

8. At this point, your installation is upgraded. Click **Continue**.
9. 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 597. For information on Cloudera Navigator, see Cloudera Navigator 2 Overview.

Upgrading from a Cloudera Enterprise Enterprise Data Hub Edition Trial to Cloudera Enterprise

1. Purchase a Cloudera Enterprise license from Cloudera.
2. On the License page, click **Upload License**.
3. Click the document icon to the left of the **Select a License File** text field.
4. Go to the location of your license file, click the file, and click **Open**.
5. Click **Upload**.

Upgrading from Cloudera Express to Cloudera Enterprise

1. Purchase a Cloudera Enterprise license from Cloudera.
2. On the License page, click **Upload License**.
3. Click the document icon to the left of the **Select a License File** text field.
4. Go to the location of your license file, click the file, and click **Open**.
5. Click **Upload**.
6. Cloudera Manager displays a pop-up describing the features enabled with Cloudera Enterprise Enterprise Data Hub Edition. Click **OK** to proceed. At this point, your installation is upgraded and the Customize Role Assignments page displays.
7. Under **Reports Manager** click **Select a host**. The pageable host selection dialog box displays.

   The following shortcuts for specifying hostname patterns are supported:

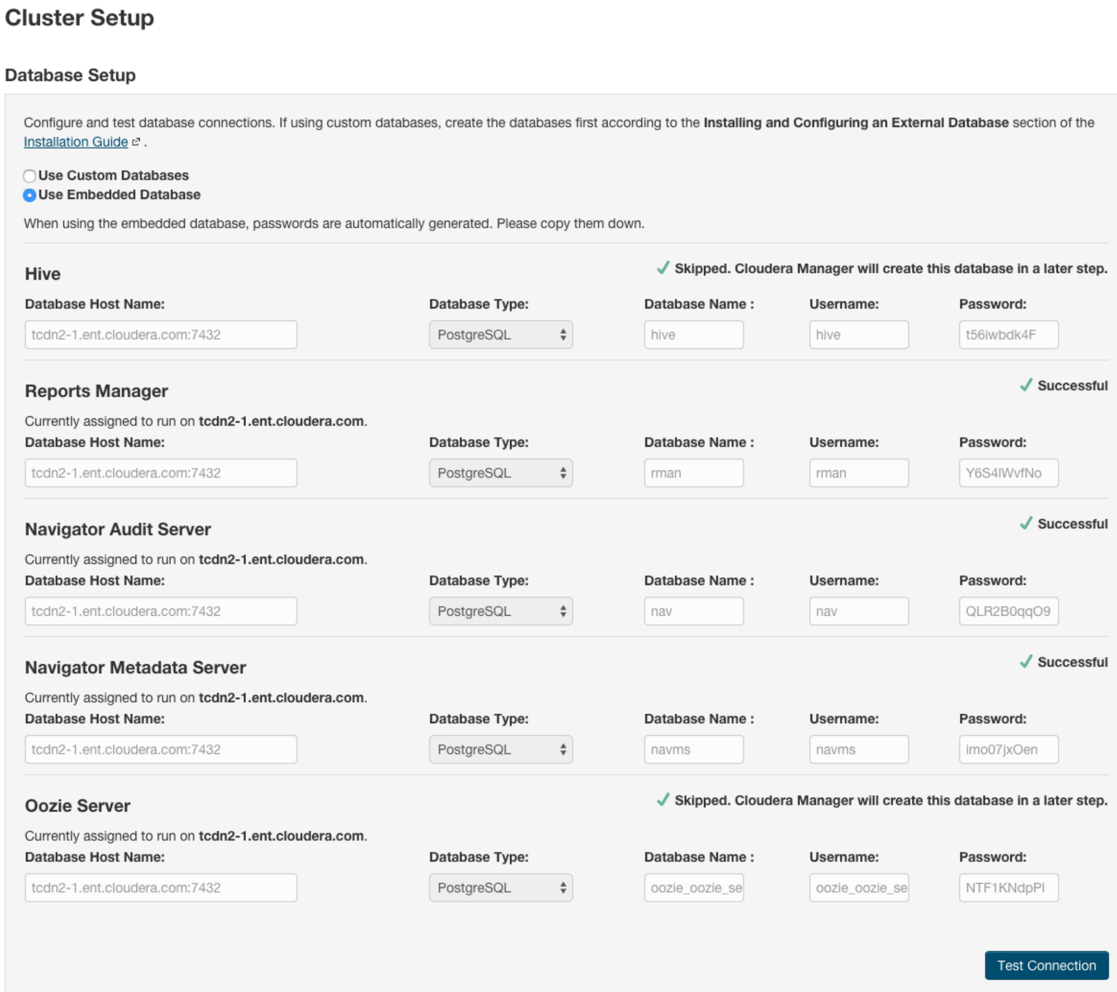
   - Range of hostnames (without the domain portion)

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

8. When you are satisfied with the assignments, click **Continue**.
9. 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.



- Select **Use Custom Databases** to specify external database host, enter the 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 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 239.

10. 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 **Cluster Setup Review Changes** screen displays.

11. 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.

12. At this point, your installation is upgraded. Click **Continue**.

13. 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 597. For information on Cloudera Navigator, see Cloudera Navigator 2 Overview.

If you want to use the Cloudera Navigator Metadata Server, add its role following the instructions in Adding Cloudera Navigator Roles on page 597.

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**.

Cloudera Manager requires a restart 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 575.

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. Clear 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 Using Cloudera Manager on page 13.

6. Click **Save Changes** to commit the changes.
7. 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 this data weekly and to send it *automatically*. Cloudera analyzes this data and uses it to improve the software. If Cloudera discovers a serious issue, Cloudera searches this diagnostic data and notifies customers with Cloudera Enterprise licenses who might encounter problems due to the issue. 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.

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 591.

Automatically sending diagnostic data might 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, and so on.
- 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. 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 or host it's running with.
- HDFS or 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.

Redaction of Sensitive Information from Diagnostic Bundles

By default, Cloudera Manager redacts known sensitive information from inclusion in diagnostic bundles. Cloudera Manager uses a set of standard rules to redeact passwords and secrets. You can add additional redaction rules using regular expressions to specify data you want to be redacted from the bundles.

To specify redaction rules for diagnostic bundles:

1. Go to **Administration** > **Settings** and search for the **Redaction Parameters for Diagnostic Bundles** parameter.

   The edit screen for the property displays.

2. To add a new rule, click the ➕ icon. You can add one of the following:

   a. **Credit Card numbers (with separator)**
   b. **Social Security Card numbers (with separator)**
   c. **Email addresses**
   d. **Custom rule** (You must supply values for the **Search** and **Replace** fields, and optionally, the **Trigger** field.)

   fields:
3. To modify a new rule, click the › icon.
4. Edit the redaction rules as needed. Each rule has a description field where you can enter free text describing the rule and you can modify the following three fields:

   - **Search -** A regular expression matched against the data. If the expression matches any part of the data, the match is replaced by the contents of the replace string. For example, to redact credit card numbers, your regular expression is `\d{4}[^\w]\d{4}[^\w]\d{4}[^\w]\d{4}`.
   - **Replace -** The string used to replace the redacted data. For example, to replace any matched credit card digits with Xs, the Replace string value would be `XXXX-XXXX-XXXX-XXXX`.
   - **Trigger -** An optional field that specifies a simple string to be searched for in the data. The redactor searches for matches to the search regular expression only if the string is found. If no trigger is specified, redaction occurs when the Search regular expression is matched. Using the Trigger field improves performance: simple string matching is faster than regular expression matching.

5. To delete a redaction rule, click the ➖ icon.
6. Click **Save 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 583.

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.

Manually Triggering Collection and Transfer of Diagnostic Data to Cloudera

To troubleshoot specific problems, or to re-send an automatic bundle that failed to send, you can manually send 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.
    - Select one of the following under **Data Selection**:
        - Select **By Target Size** to manually set the maximum size of the bundle. 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.
        - Select **By Date Range** to manually set the **Start Time** and **End Time** to collect the diagnostic data. Click the **Estimate** button to calculate the size of the bundle based on the start and end times. If the bundle is too large, narrow the selection using the start and end times or by selecting additional filters.
    - 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.

Backing up Cloudera Manager

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

The following steps create a complete backup of Cloudera Manager:

1. Stop the Cloudera Management Service using Cloudera Manager:
   a. Select **Clusters** > **Cloudera Management Service**.
   b. Select **Actions** > **Stop**.
2. Stop the Cloudera Manager Server by running the following command on the Cloudera Manager Server host:

```
sudo service cloudera-scm-server stop
```

3. If you are using the embedded PostgreSQL database for Cloudera Manager, stop the database:

```
sudo service cloudera-scm-server-db stop
```

> **Important:** If you are *not* running the embedded database service and you attempt to stop it, you receive a message indicating that the service cannot be found. If instead you get a message that the shutdown failed, the embedded database is still running, probably because services are connected to the Hive metastore. If the database shutdown fails due to connected services, issue the following command:
>
> - RHEL-compatible 7 and higher:
>
> ```
> $ sudo service cloudera-scm-server-db next_stop_fast
> $ sudo service cloudera-scm-server-db stop
> ```
>
> - All other Linux distributions:
>
> ```
> sudo service cloudera-scm-server-db fast_stop
> ```

4. On the host where Cloudera Manager Server is running, back up the `/etc/cloudera-scm-server/db.properties` file.
5. On the host where each role is running, back up the following directories whose location is stored in the properties shown in the table. The default location is also shown.

Table 37: Cloudera Management Service Directories to Back Up

Role	Property	Default Location
Event Server	**Event Server Index Directory**	`/var/lib/cloudera-scm-eventserver`
Host Monitor	**Host Monitor Storage Directory**	`/var/lib/cloudera-host-monitor`
Navigator Metadata Server	**Navigator Metadata Server Storage Dir**	`/var/lib/cloudera-scm-navigator`
Reports Manager	**Reports Manager Working Directory**	`/var/lib/cloudera-scm-headlamp`
Service Monitor	**Service Monitor Storage Directory**	`/var/lib/cloudera-service-monitor`

6. Back up the `/etc/cloudera-scm-agent/config.ini` file on each host in the cluster.
7. Back up the following Cloudera Manager-related databases; see Backing up Databases:
   - Cloudera Manager Server
   - Activity Monitor (depending on your deployment, this role may not exist)
   - Reports Manager
   - Navigator Audit Server
   - Navigator Metadata Server
8. Start Cloudera Manager Server by running the following command on the Cloudera Manager Server host:

```
sudo service cloudera-scm-server start
```

9. Start the Cloudera Management Service using Cloudera Manager:
   a. Select **Clusters** > **Cloudera Management Service**.
   b. Select **Actions** > **Start**.

Backing up Databases

Several steps in the backup procedures require you to back up various databases used in a CDH cluster. The steps for backing up and restoring databases differ depending on the database vendor and version you select for your cluster and are beyond the scope of this document.

See the following vendor resources for more information:

- **MariaDB 5.5:** http://mariadb.com/kb/en/mariadb/backup-and-restore-overview/
- **MySQL 5.5:** http://dev.mysql.com/doc/refman/5.5/en/backup-and-recovery.html
- **MySQL 5.6:** http://dev.mysql.com/doc/refman/5.6/en/backup-and-recovery.html
- **PostgreSQL 8.4:** https://www.postgresql.org/docs/8.4/static/backup.html
- **PostgreSQL 9.2:** https://www.postgresql.org/docs/9.2/static/backup.html
- **PostgreSQL 9.3:** https://www.postgresql.org/docs/9.3/static/backup.html
- **Oracle 11gR2:** http://docs.oracle.com/cd/E11882_01/backup.112/e10642/toc.htm

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 569.
- **Advanced** - Enable API debugging and other advanced options.
- **Monitoring** - Set Agent health status parameters. For configuration instructions, see Configuring Cloudera Manager Agents on page 569.
- **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 559.
- **Other**
  - Enable Cloudera usage data collection For configuration instructions, see Managing Anonymous Usage Data Collection on page 588.
  - 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 589.
  - 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 575.
- **Custom Service Descriptors** - Configure custom service descriptor properties for Add-on Services on page 42.

Alerts

See Managing Alerts on page 576.

Users

See Cloudera Manager User Accounts.

Kerberos

See Enabling Kerberos Authentication Using the Wizard.

License

See Managing Licenses on page 583.

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 478.

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 **Home** > **Status** tab, 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** > **Status** tab, 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** > **Status** tab, 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** > **Status** tab, 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 **Home** > **Status** tab, 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 **Home** > **Status** tab, 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 Using Cloudera Manager on page 13.
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 **Home** > **Status** tab, 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 box 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 box.

      The following shortcuts for specifying hostname patterns are supported:

      - Range of hostnames (without the domain portion)

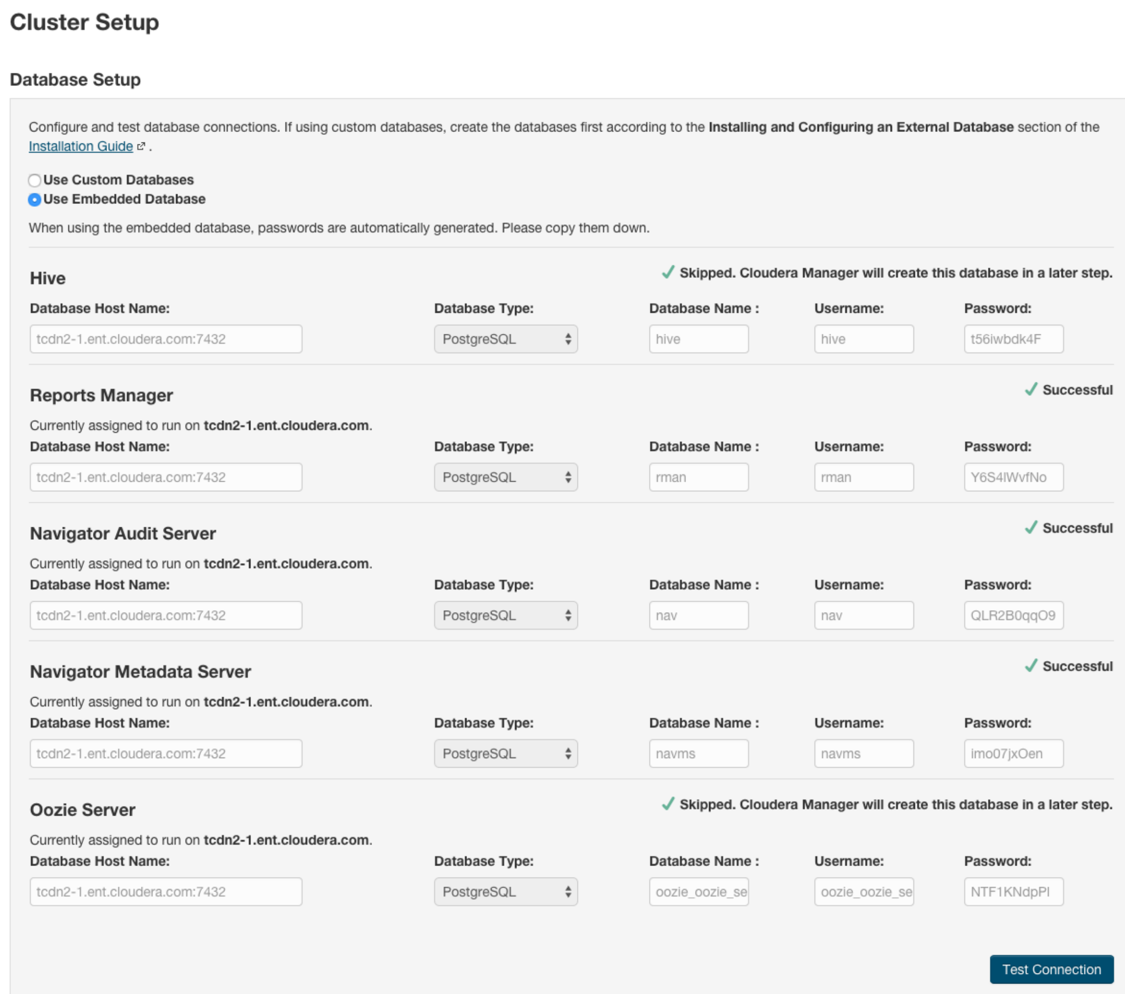
Range Definition	Matching Hosts
10.1.1.[1-4]	10.1.1.1, 10.1.1.2, 10.1.1.3, 10.1.1.4
host[1-3].company.com	host1.company.com, host2.company.com, host3.company.com

Range Definition	Matching Hosts
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. 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.



   - Select **Use Custom Databases** to specify external database host, enter the 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 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 239.

7. 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 **Cluster Setup Review Changes** screen displays.

8. 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 **Home** > **Status** tab, 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 595.

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 TLS/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 583.

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 **Home** > **Status** tab, 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 box 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 box.

      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

Range Definition	Matching Hosts
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

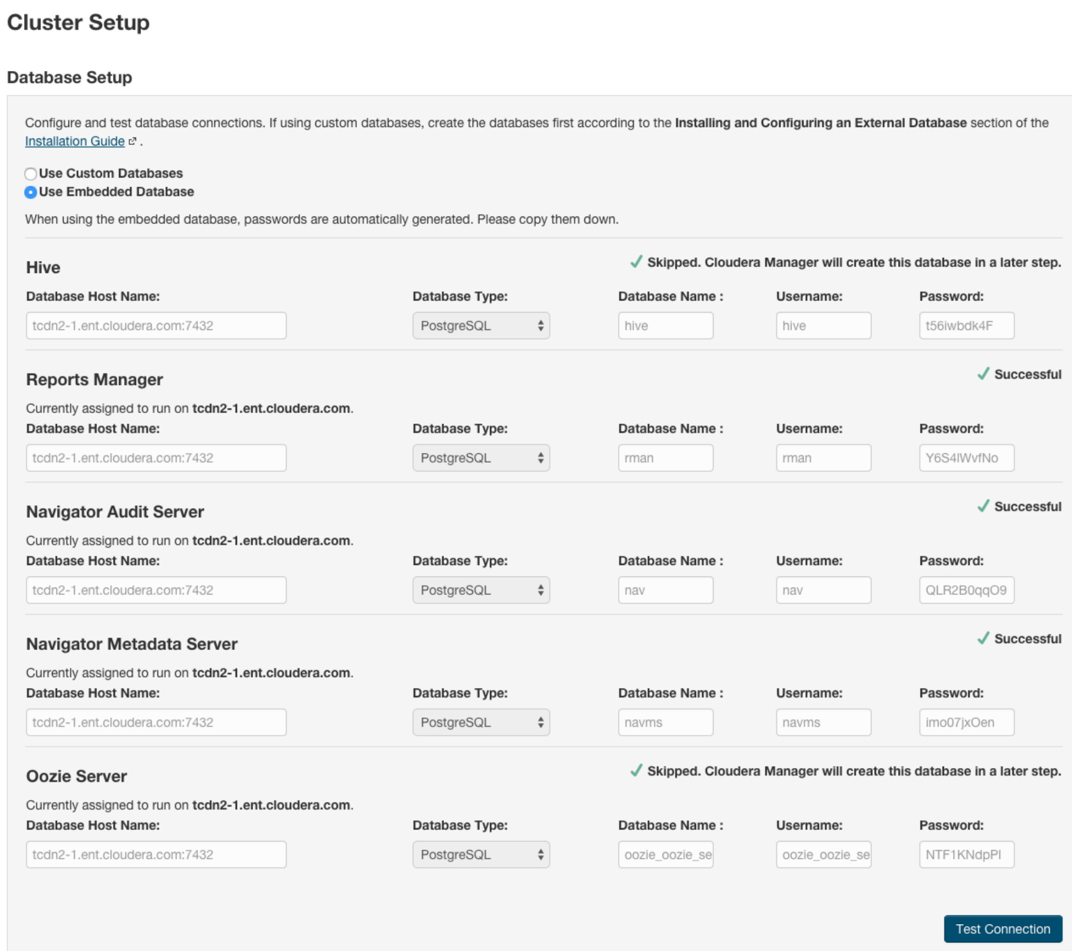
- IP addresses
- Rack name

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.



- 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.

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 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 stating that the database will be created later 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 **Home** > **Status** tab, 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 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 **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Category** > **All**.
4. 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 Using Cloudera Manager on page 13.
5. Restart the role.

Configuring the Navigator Audit Server Log Directory

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

You can configure the location of the Navigator Audit Server logs (by default, `/var/log/cloudera-scm-navigator`) as follows:

1. Do one of the following:
    - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
    - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.

3. Select **Category** > **Logs**.
4. 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 Using Cloudera Manager on page 13.

5. Restart the role.

Configuring Service Audit Collection and Log Properties

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

To configure service auditing, you must enable audit collection and log properties. You can also configure which events are logged for each service by following the procedures in Configuring Audit Properties on page 605.

Enabling Audit Collection

The service or role **Enable Audit Collection** property controls whether the Cloudera Manager Agent tracks a service or role's audit log file.

1. Do one of the following:
   - Go to a supported service.
   - Go to Navigator Metadata Server.
     - Do one of the following:
       - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
       - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select the scope according to the service:
   - Service - **Scope** > ***ServiceName*** **(Service-Wide)**
   - Navigator Metadata Server - **Scope** > **Navigator Metadata Server**
4. Select **Category** > **Cloudera Navigator**.
5. Select the **Enable Audit Collection** checkbox.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

Configuring Impala Daemon Logging

To control whether the Impala Daemon role logs to the audit log:

1. Click the Impala service.
2. Click the **Configuration** tab.
3. Select **Scope** > **Impala Daemon**.
4. Select **Category** > **Logs**.
5. Edit the **Enable Impala Audit Event Generation**.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

To set the log file size:

1. Click the Impala service.
2. Select **Scope** > **Impala Daemon**.
3. Select **Category** > **Logs**.
4. Set the **Impala Daemon Maximum Audit Log File Size** property.

5. Click **Save Changes** to commit the changes.
6. Restart the service.

Enabling Solr Auditing

Solr auditing is disabled by default. To enable auditing:

1. Enable Sentry authorization for Solr following the procedure in Enabling Sentry Policy File Authorization for Solr.
2. Go to the Solr service.
3. Click the **Configuration** tab.
4. Select **Scope** > ***Solr Service*** **(Service-Wide)**
5. Select **Category** > **Policy File Based Sentry** category.
6. Select or clear the **Enable Sentry Authorization** checkbox.
7. Select **Category** > **Cloudera Navigator** category.
8. Select or clear the **Enable Audit Collection** checkbox. See Configuring Service Audit Collection and Log Properties on page 603.
9. Click **Save Changes** to commit the changes.
10. Restart the service.

Configuring Audit Logs

The following properties apply to an audit log file:

- **Audit Log Directory** - The directory in which audit log files are written. By default, this property is not set if Cloudera Navigator is not installed.

  A validation check is performed for all lifecycle actions (stop/start/restart). If the Enable Collection flag is selected and the Audit Log Directory property *is not set*, the validator displays a message that says that the Audit Log Directory property must be set to enable auditing.

  If the value of this property is changed, and service is restarted, then the Cloudera Manager Agent will start monitoring the new log directory for audit events. In this case it is possible that not all events are published from the old audit log directory. To avoid loss of audit events, when this property is changed, perform the following steps:

  1. Stop the service.
  2. Copy audit log files and (for Impala only) the `impalad_audit_wal` file from the old audit log directory to the new audit log directory. This needs to be done on all the hosts where Impala Daemons are running.
  3. Start the service.

- **Maximum Audit Log File Size** - The maximum size of the audit log file before a new file is created. The unit of the file size is service dependent:

  – **HDFS, HBase, Hive, Hue, Navigator Metadata Server, Sentry, Solr** - MiB
  – **Impala** - lines (queries)

- **Number of Audit Logs to Retain** - Maximum number of rolled over audit logs to retain. The logs will not be deleted if they contain audit events that have not yet been propagated to the Audit Server.

To configure audit logs do the following:

1. Do one of the following:

   - Service - Go to a supported service.
   - Navigator Metadata Server

     – Do one of the following:

       – Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
       – On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the **Configuration** tab.
3. Select the scope according to the service:
   - All services except Impala - **Scope** > ***ServiceName*** **(Service-Wide)**.
   - Impala - **Scope** > **Impala Daemon**.
   - Navigator Metadata Server - **Scope** > **Navigator Metadata Server**.
4. Select **Category** > **Logs**.
5. Configure the log properties. For Impala, preface each log property with **Impala Daemon**.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

Configuring Audit Properties

Set auditing properties for a service in the service's configuration settings in Cloudera Manager.

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

Setting Audit Properties

To review or update audit properties for a service:

1. Log in to the Cloudera Manager Admin Console.
2. Select select any service that supports auditing. Select **Cluster** > ***Service-Name***
3. Click the **Configuration** tab.
4. Select ***Service*** **(Service-Wide)** for the **Scope** filter.
5. Select **Cloudera Navigator** for the **Category** filter.
6. Edit the properties.
7. Click **Save Changes** to commit the changes.
8. Restart the service if needed.

   It is not necessary to restart the service if you are updating an audit filter.

In addition, for Impala and Solr auditing, perform the steps in Configuring Impala Daemon Logging on page 603 and Enabling Solr Auditing on page 604 and for all services perform the steps in Configuring Audit Logs on page 604.

Operations within Navigator are also audited. To update Navigator's audits:

1. Select **Clusters** > **Cloudera Management Service**.
2. Click the **Configuration** tab.
3. Select **Navigator Metadata Server** for the **Scope** filter.
4. Select **Category** > **Cloudera Navigator** category.
5. Edit the properties.
6. Click **Save Changes** to commit the changes.
7. Restart the service if needed.

   It is not necessary to restart the service if you are updating an audit filter.

Service-level Audit Properties

Each service (with exceptions noted) that supports auditing has the following properties:

- **Enable Audit Collection** - See Enabling Audit Collection on page 603 and Enabling Solr Auditing on page 604.
- **Audit Log properties** - See Configuring Audit Logs on page 604 and Configuring Impala Daemon Logging on page 603.
- **Audit Event Filter** - A set of rules that capture properties of auditable events and actions to be performed when an event matches those properties. The Cloudera Manager Agent uses this property to filter events out *before* they are sent to Cloudera Navigator. The default filter settings discard the following events:
  - **HDFS**

    - Generated by the internal Cloudera and Hadoop users (`cloudera-scm`, `hdfs`, `hbase`, `hive`, `impala`, `mapred`, `solr`, `spark`, and `dr.who`)
    - Generated by the `hdfs` user running the listStatus, listCachePools, listCacheDirectives, and getfileinfo operations
    - That affect files in the `/tmp` directory
  - **HBase** - That affect the -ROOT-, .META., and acl tables
  - **Hive** - Generated by Hive MapReduce jobs in the /tmp directory
  - **Impala**, **Solr**, **Solr**, **Navigator Metadata Server** - No default filter
- **Audit Event Tracker** - A set of rules for tracking and coalescing events. This feature is used to define equivalency between different audit events. Tracking works by keeping a reference to events when they first appear, and comparing other incoming events against the tracked events according to the rules defined. When events match, according to a set of configurable parameters, only one entry in the audit list is generated for all the matching events. This property is not supported for the Navigator Metadata Server.
- **Audit Queue Policy** - The action to take when the audit event queue is full. The options are Drop or Shutdown. When a queue is full and the queue policy of the service is Shutdown, before shutting down the service, *N* audits will be discarded, where *N* is the size of the Cloudera Navigator Audit Server queue.

  Note: If the queue policy is Shutdown, the Impala service is shut down only if Impala is unable to write to the audit log file. It is possible that an event may not appear in the audit event log due to an error in transfer to the Cloudera Manager Agent or database. In such cases Impala will not shut down and will keep writing to the log file. When the transfer problem is fixed the events will be transferred to the database.

  This property is not supported for Hue or the Navigator Metadata Server.

Finding Audit Filter Properties for Each Service

For information on the structure of the objects, and the properties for which you can set filters, display the description on the configuration page as follows:

1. Log in to the Cloudera Manager Admin Console.
2. Select select any service that supports auditing. Select **Cluster** > ***Service-Name***
3. Click the **Configuration** tab.
4. Select ***Service*** **(Service-Wide)** for the **Scope** filter.
5. Select **Cloudera Navigator** for the **Category** filter.
6. Click for **Audit Event Tracker** to open the property description.

   For example, the Hive properties are:

   - operation: the Hive operation being performed.
   - userName: the user performing the action.
   - ipAddress: the IP from where the request originated.
   - allowed: whether the operation was allowed or denied.
   - databaseName: the databaseName for the operation.
   - tableName: the tableName for the operation.
   - objectType: the type of object affected by the operation.
   - resourcePath: the path of the resource affected by the operation.

Entering Audit Filter Properties

The Audit Event Filter and Audit Event Tracker rules for filtering and coalescing events are expressed as JSON objects. You can edit these rules using a rule editor. The `username` filters shown here include an optional phrase (`(?:/.+)?`) to filter events where the user name is constructed with a slash as part of the name, such as a Kerberos principal (`username/instance@REALM`). It applies only if the regular expression matches one of the usernames listed in the first phrase. The `src` filter shows a similar optional phrase (`(?:/.*)?`) that filters for subdirectories under the `/tmp` directory. The regular expression looks for occurrences of slash followed by zero or more characters; this makes sure that subdirectories are included but other directories whose name starts with "tmp" are not included in the filter.

HDFS-1 (Service-Wide) View as JSON

- Action: discard Fields: username: (?:cloudera-scm|hbase|mapred|hive|dr.who|solr|impala|spark)(?:/.+)?
- Action: discard Fields: username: (?:hdfs)(?:/.+)?, operation: (?:listStatus|listCachePools|listCacheDirectives|getfileinfo)
- Action: discard Fields: src: /tmp(?:/.*)?

Default action: Accept

or in a JSON text field:

HDFS-1 (Service-Wide) View Editor


```
{
  "rules": [
    {
      "action": "discard",
      "fields": [
        {
          "name": "username",
          "match": "(?:cloudera-scm|hbase|mapred|hive|dr.who|solr|impala|spark)(?:/.+)?"
        }
      ]
    },
    {
      "action": "discard",
      "fields": [
        {
          "name": "username",
          "match": "(?:hdfs)(?:/.+)?"
        },
        {
          "name": "operation",
          "match": "(?:listStatus|listCachePools|listCacheDirectives|getfileinfo)"
        }
      ]
    },
    {
      "action": "discard",
      "fields": [
        {
          "name": "src",
          "match": "/tmp(?:/.*)?"
        }
      ]
    }
  ],
  "defaultAction": "accept",
  "comment": [
    "Default filter for HDFS services.",
    "Discards events generated by the internal Cloudera and/or HDFS users",
    "(cloudera-scm, hbase, mapred, hive, dr.who, solr, impala, and spark),",
    "'ls' actions performed by the hdfs user,",
    "and events that affect files in the /tmp directory."
  ]
}
```


Audit Filter Examples

Use audit filters to control what information is collected by Navigator Audit Server. The following sections describe cases where a filter can allow you to remove noise from the audit events or otherwise improve the quality of the audit trail.

Removing audit events for automated tasks

If you find that the audit events include references to automated tasks against HDFS or other services that you don't want to include in your audit, you can include the tasks in a "discard" filter. For example, to discard all tasks performed against HDFS by the user under which the automated tasks are performed, create an HDFS filter:

```
Action: Discard
Fields: username: (?:automationuser|automationuser@realm.example.com)
```

This rule would discard any event performed by the `automationuser` account, where the user is specified by itself or with a Kerberos realm designation. Note that the filter is not case sensitive. As a performance optimization, the expression is included in parentheses with `?:` to indicate that the matched values are not retained.

Removing audit events for low priority operations

You may find that your audit log includes reference to events that you are not interested in tracking. You can prevent these events from being included in the audit log by filtering them out by operation. For example, if the audit event that you don't want to track looks like this:

```
2170816,hdfs,1,someuser@MYREALM.COM,null,192.168.12.18,1477605498525,getfileinfo,
/user/someuser/.staging/jobx/status,null,null,null
```

where the HDFS operation `getfileinfo` ran against the file `status` for the user `someuser` on a node in the cluster. The user account is indicated with a Kerberos realm.

To remove the HDFS getfileinfo accesses by this user, the audit filter would reference the user (by itself or with a Kerberos realm name) and the getfileinfo operation:

```
Action: Discard
Fields: username: (?:someuser|someuser@MYREALM.com)
        operation: (?:getfileinfo)
```

As a performance optimization, the expressions are included in parentheses with `?:` to indicate that the matched values are not retained.

To include more than one operation in the same filter, include both operations with an OR operator `|.`:

```
Action: Discard
Fields: username: (?:someuser|someuser@MYREALM.com)
        operation: (?:getfileinfo|liststatus)
```

Publishing Audit Events

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

You can publish audit events to a Kafka topic or syslog.

A failure to send an event to Kafka or syslog is logged to the Audit Server log.

Publishing Audit Events to Kafka

You can publish audit events to a Kafka topic. To configure audit logging to a Kafka topic, do the following:

1. Do one of the following:
   - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
   - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Type "kafka" in the Search box to display the configurable settings.
4. For the **Kafka Service** property, select the Kafka service to which Cloudera Navigator will publish audit events.

5. For the **Kafka Topic** property, enter the name of the default group to which Cloudera Navigator will publish the audit events.
6. Click **Save Changes**.
7. Restart the role.

Publishing Audit Events to Syslog

The Audit Server logs all audit records into a Log4j logger called `auditStream`. The log messages are logged at the TRACE level, with the attributes of the audit records. By default, the `auditStream` logger is inactive because the logger level is set to FATAL. It is also connected to a `NullAppender`, and does not forward to other appenders (additivity set to false).

To record the audit stream, configure the `auditStream` logger with the desired appender. For example, the standard `SyslogAppender` allows you to send the audit records to a remote syslog.

The Log4j `SyslogAppender` supports only UDP. An example syslog configuration would be:

```
$ModLoad imudp
$UDPServerRun 514
# Accept everything (even DEBUG messages)
local2.* /my/audit/trail.log
```

It is also possible to attach other appenders to the `auditStream` to provide other integration behaviors.

You can publish audit events to syslog in two formats: JSON and RSA EnVision. To configure audit logging to syslog, do the following:

1. Do one of the following:
    - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
    - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Locate the **Navigator Audit Server Logging Advanced Configuration Snippet** property by typing its name in the Search box.
4. Depending on the format type, enter:

```
log4j.appender.SYSLOG = org.apache.log4j.net.SyslogAppender
log4j.appender.SYSLOG.SyslogHost = hostname
log4j.appender.SYSLOG.Facility = Local2
log4j.appender.SYSLOG.FacilityPrinting = true
```

   To configure the specific stream type, enter:

Format	Property
JSON	`log4j.logger.auditStream = TRACE,SYSLOG` `log4j.additivity.auditStream = false`
RSA EnVision	`log4j.logger.auditStreamEnVision = TRACE,SYSLOG` `log4j.additivity.auditStreamEnVision = false`

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Example Log Messages

Format	Log Message Example

JSON	`Jul 23 11:05:15 hostname local2:`
	`{"type":"HDFS","allowed":"true","time":"1374602714758",`
	`"service":"HDFS-1",`
	`"user":"root","ip":"10.20.93.93","op":"mkdirs","src":"/audit/root","perms":"rwxr-xr-x"}`

Format	Log Message Example
RSA EnVision	`Cloudera|Navigator|1|type="Hive",allowed="false",time="1382551146763", service="HIVE-1",user="systest",impersonator="",ip="/10.20.190.185",op="QUERY", opText="select count(*) from sample_07",db="default",table="sample_07",path="/user/hive/warehouse/sample_07",objType="TABLE"`

If a particular field is not applicable for that audit event, it is omitted from the message.

Cloudera Navigator Metadata Server

The Navigator Metadata Server performs the following functions:

- Obtains connection information about CDH services from the Cloudera Manager Server
- At periodic intervals, extracts metadata for the entities managed by those services
- Manages and applies metadata extraction policies during metadata extraction
- Indexes and stores entity metadata
- Manages authorization data for Cloudera Navigator users
- Manages audit report metadata
- Generates metadata and audit analytics
- 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 583.

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 **Home** > **Status** tab, 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 box 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 box.

      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

Range Definition	Matching Hosts
host[07-10].company.com	host07.company.com, host08.company.com, host09.company.com, host10.company.com

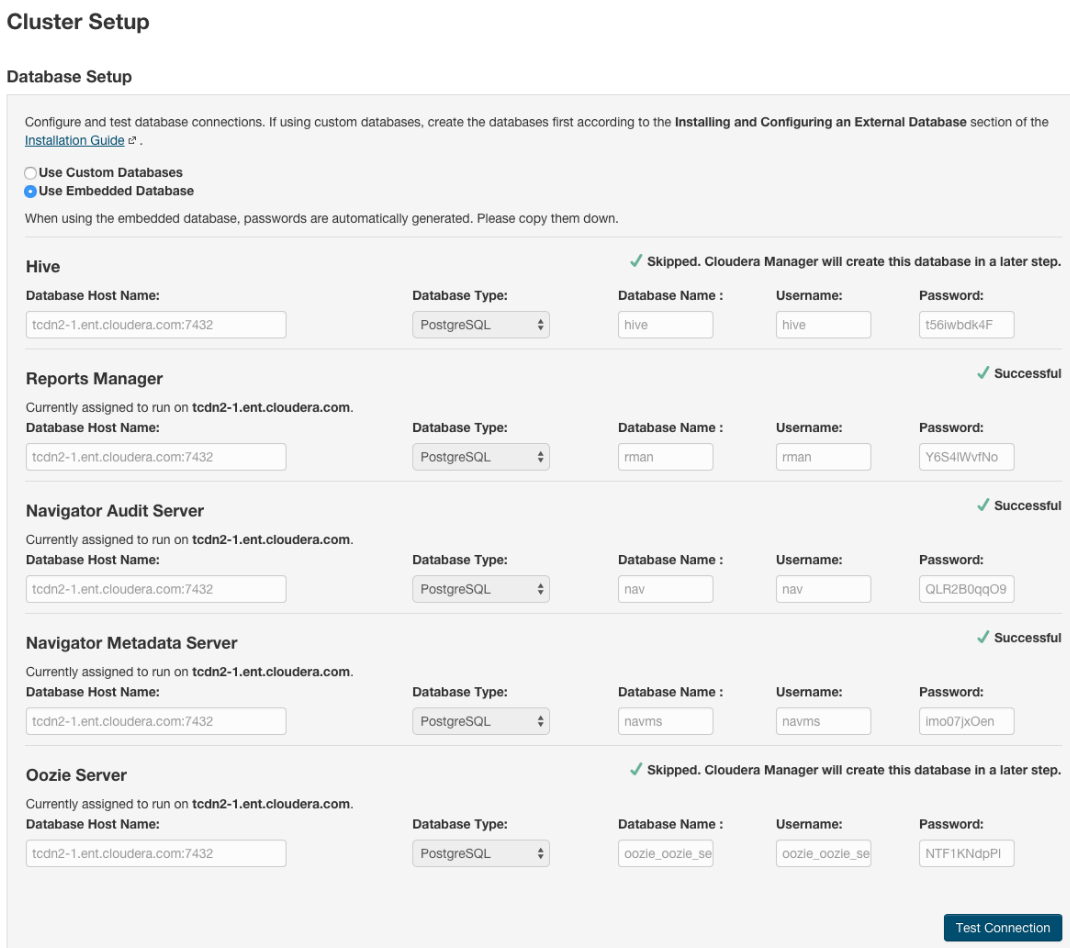
- IP addresses
- Rack name

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.



- Select **Use Custom Databases** to specify external database host, enter the 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 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 239.

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 **Cluster Setup Review Changes** screen displays.

8. Click **Finish**.

Starting, Stopping, and Restarting the Navigator Metadata Server

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 **Home** > **Status** tab, 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 Memory Sizing 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. The Navigator Metadata Server memory requirements are based on amount of data that is stored and indexed. The Solr instance runs in process with Navigator, so you should set Java heap for the Navigator Metadata Server accordingly.

When the Navigator Metadata Server starts up it logs the number of documents contained in Solr. For example:

```
2015-11-11 09:24:58,013 INFO com.cloudera.nav.server.NavServerUtil: Found 68813088
documents in solr core nav_elements
2015-11-11 09:24:58,705 INFO com.cloudera.nav.server.NavServerUtil: Found 78813930
documents in solr core nav_relations
```

To compute the memory required by the Metadata Server during *normal operation*, use the number of documents in `nav_elements * 200`. So for the above example, the recommended amount of memory would be `(68813088 * 200)` or about 14 GB.

If you are *upgrading* the Navigator Metadata Server, use the number of documents in `nav_elements + nav_relations`. If you use the number in the example above, you would need `((68813088 + 78813930) * 200)`, or about 30 GB, to upgrade Navigator Metatdata Server.

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.

Setting the Navigator Metadata Server Java Heap Size

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 **Home** > **Status** tab, 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 memory value in the **Java Heap Size of Navigator Metadata Server in Bytes** property.
5. Click **Save Changes** to commit the changes.
6. Click the **Instances** tab.
7. Restart the role.

Moving the Navigator Metadata Server to a New Host

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.
2. If the storage directory 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 Navigator Metadata Server stores extracted and indexed data in a directory on the host on which the Server runs. The default is `/var/lib/cloudera-scm-navigator`. To configure the storage directory:

1. Do one of the following:
   - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
   - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Instances** tab.
3. Stop the Navigator Metadata Server.
4. In a terminal window, log into the host on which the Navigator Metadata Server is running and copy the current storage directory to the new location.
5. Click the **Configuration** tab.
6. Select **Scope** > **Navigator Metadata Server**.
7. 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 Using Cloudera Manager on page 13.
8. Click **Save Changes** to commit the changes.
9. Click the **Instances** tab.
10. Restart the role.

Configuring the Navigator Metadata Server Port

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

You can set 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 **Home** > **Status** tab, 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 Using Cloudera Manager on page 13.
5. Click **Save Changes** to commit the changes.
6. Click the **Instances** tab.
7. Restart the role.

Configuring the Navigator Metadata Server Log Directory

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

You can configure the location of the Navigator Metadata Server logs (by default, `/var/log/cloudera-scm-navigator`) as follows:

1. Do one of the following:
    - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
    - On the **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.
2. Click the **Configuration** tab.
3. Select **Scope** > **Navigator Metadata Server**.
4. Select **Category** > **Logs**.
5. Set the **Navigator Metadata 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 Using Cloudera Manager on page 13.
6. Click **Save Changes** to commit the changes.
7. Restart the role.

Configuring the Navigator Metadata Server 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 **Home** > **Status** tab, 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 Using Cloudera Manager on page 13.

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 uses Google Analytics to send anonymous usage information to Cloudera. The information helps Cloudera improve 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 **Home** > **Status** tab, 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 clear 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 Using Cloudera Manager on page 13.
5. Click **Save Changes** to commit the changes.
6. Restart the role.

Managing Hive and Impala Lineage Properties

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

For Hive and Impala, query information is not extracted by the Navigator Metadata Server. Instead, query information is written by the services to log files. The directory containing the log files is monitored by the Cloudera Manager Agent. Periodically the log files are collected by the Cloudera Manager Agent and forwarded to the Navigator Metadata Server.

Enabling and Disabling Hive and Impala Lineage

The **Enable Lineage Collection** property determines whether lineage logs should be collected by the Cloudera Manager Agent. Both Hive and Impala lineage collection is enabled by default. To control whether the Impala Daemon role logs to the lineage log and whether the Cloudera Manager Agent collects the Hive and Impala lineage entries:

1. Go to the Hive or Impala service.
2. Click the **Configuration** tab.
3. Type `lineage` in the Search box.
4. Select or clear the **Enable Lineage Collection** checkbox.
5. (Impala only) Select or clear the **Enable Impala Lineage Generation** checkbox.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

If you clear *either* Impala checkbox, Impala lineage is disabled.

Configuring Hive on Spark and Impala Daemon Lineage Logs

The following properties apply to the Hive and Impala lineage log files:

- **Hive Lineage Log Directory** - The directory in which Hive lineage log files are written.
- **Hive Maximum Lineage Log File Size** - The maximum size in MB of the Hive lineage log file before a new file is created.

- **Enable Impala Lineage Generation** - Indicates whether Impala lineage logs should be generated.
- **Impala Daemon Lineage Log Directory** - The directory in which Impala lineage log files are written.
- **Impala Daemon Maximum Lineage Log File Size** - The maximum size in number of queries of the lineage log file before a new file is created.

If the value of a log directory property is changed, and service is restarted, the Cloudera Manager Agent starts monitoring the new log directory. In this case it is possible that not all events are published from the old directory. To avoid losing lineage information when this property is changed, perform the following steps:

1. Stop the affected service.
2. Copy the lineage log files and (for Impala only) the `impalad_lineage_wal` file from the old log directory to the new log directory. This needs to be done on the HiveServer2 host and all the hosts where Impala Daemon roles are running.
3. Start the service.

To edit lineage log properties:

1. Go to the service.
2. Click the **Configuration** tab.
3. Type `lineage` in the Search box.
4. Edit the lineage log properties.
5. Click **Save Changes** to commit the changes.
6. Restart the service.

Managing Metadata

This topic describes tasks for enabling and disabling metadata extraction and purging obsolete metadata.

Enabling and Disabling Metadata Extraction

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

Enabling Hive Metadata Extraction in a Secure Cluster

The Navigator Metadata Server uses the hue user to connect to the Hive Metastore. The hue user can connect to the Hive Metastore by default. However, if the Hive service **Hive Metastore Access Control and Proxy User Groups Override** property 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 cannot extract metadata from Apache Hive. If this is the case, modify the Hive service **Hive Metastore Access Control and Proxy User Groups Override** property or the HDFS service **Hive Proxy User Groups** property 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 Using Cloudera Manager on page 13.
5. Type hue.
6. Click **Save Changes** to commit the changes.
7. Restart the service.

Enabling Spark Metadata Extraction

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 **Home** > **Status** tab, 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 Using Cloudera Manager on page 13.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Managing Metadata Capacity

Minimum Required Role: Full Administrator

The metadata maintained by Navigator Metadata Server can grow rapidly and exceed the capacity of the Solr instance storing the data. Navigator Metadata Server purge allows you to delete unwanted metadata to improve performance and reduce noise during search and lineage. Currently purge is available only through the Metadata Server API.

Purging Metadata for HDFS Entities, Hive and Impala Select Queries, and YARN, Sqoop, and Pig Operations

You can delete metadata for HDFS entities, Hive and Impala select queries, and YARN, Sqoop and Pig operations by using the `purge` method. (Metadata for Hive tables is not deleted.) Purge is a long-running task that requires exclusive access to the Solr instance and does not allow any concurrent activities, including extraction.

To purge metadata, do the following:

1. Back up the Navigator Metadata Server storage directory.
2. Invoke the `http://Navigator_Metadata_Server_host:port/api/v9/maintenance/purge` endpoint with the following parameters:

Table 38: Purge Parameters

Metadata	Parameter	Description
HDFS	`deleteTimeThresholdMinutes`	After an HDFS entity is deleted, the number of minutes that must pass before that entity can be purged. **Default:** 86400 minutes (60 days).
	`runtimeCapMinutes`	Number of minutes that the HDFS purge can run. When this limit is reached, the purge state is saved and the purge task terminates. You must run the purge again to purge any remaining entities. If you set the value to 0, no HDFS files or directories are purged. **Default:** 720 minutes (12 hours).
Hive and Impala Select Queries; YARN, Sqoop, Pig Operations	`deleteSelectOperations`	Boolean. If set to true, the purge deletes all Hive and Impala select queries, and YARN, Sqoop, and Pig operations, that are older than the number of days defined by the `staleQueryThresholdDays` value. **Default:** false

Metadata	Parameter	Description
	staleQueryThresholdDays	For Hive and Impala select queries, and YARN, Sqoop, and Pig operations, the number of days they must be older than to be purged. To disable purge for Hive and Impala select queries, and for YARN, Sqoop, and Pig operations, set the threshold to a very large value, for example, 36500. **Default:** 60 days

For example, the following call purges the metadata of all deleted HDFS entities because the elapsed minutes value is set to 0:

```
$ curl -X POST -u admin:admin
"http://Navigator_Metadata_Server_host:port/api/v9/maintenance/purge?deleteTimeThresholdMinutes=0"
```

Purge tasks do not start until all currently running extraction tasks finish.

3. When all tasks have completed, click **Continue** to return to the Cloudera Navigator UI.

Retrieving Purge Status

To view the status of the purge process, invoke the `http://Navigator_Metadata_Server_host:port/api/v9/maintenance/running` endpoint. For example:

```
curl -X GET -u admin:admin
"http://Navigator_Metadata_Server_host:port/api/v9/maintenance/running"
```

A result would look similar to:

```
[{
  "id" : 5,
  "type" : "PURGE",
  "startTime" : "2016-03-10T23:17:41.884Z",
  "endTime" : "1970-01-01T00:00:00.000Z",
  "status" : "IN_PROGRESS",
  "message" : "Purged 2661984 out of 4864714 directories. Averaging 1709 directories
per minute.",
  "username" : "admin",
  "stage" : "HDFS_DIRECTORIES",
  "stagePercent" : 54
}]
```

Retrieving Purge History

To view the purge history, invoke the `http://Navigator_Metadata_Server_host:port/api/v9/maintenance/history` endpoint with the following parameters:

Table 39: History Parameters

Parameter	Description
`offset`	First purge history entry to retrieve. Default: 0.
`limit`	Number of history entries to retrieve from the offset. Default: 100.

For example:

```
curl -X GET -u admin:admin
"http://Navigator_Metadata_Server_host:port/api/v9/maintenance/history?offset=0&limit=100"
```

A result would look similar to:


```
[
  {
    "id": 1,
    "type": "PURGE",
    "startTime": "2016-03-09T18:57:43.196Z",
    "endTime": "2016-03-09T18:58:33.337Z",
    "status": "SUCCESS",
    "username": "admin",
    "stagePercent": 0
  },
  {
    "id": 2,
    "type": "PURGE",
    "startTime": "2016-03-09T19:47:39.401Z",
    "endTime": "2016-03-09T19:47:40.841Z",
    "status": "SUCCESS",
    "username": "admin",
    "stagePercent": 0
  },
  {
    "id": 3,
    "type": "PURGE",
    "startTime": "2016-03-10T01:27:39.632Z",
    "endTime": "2016-03-10T01:27:46.809Z",
    "status": "SUCCESS",
    "username": "admin",
    "stagePercent": 0
  },
  {
    "id": 4,
    "type": "PURGE",
    "startTime": "2016-03-10T01:57:40.461Z",
    "endTime": "2016-03-10T01:57:41.174Z",
    "status": "SUCCESS",
    "username": "admin",
    "stagePercent": 0
  },
  {
    "id": 5,
    "type": "PURGE",
    "startTime": "2016-03-10T23:17:41.884Z",
    "endTime": "2016-03-10T23:18:06.802Z",
    "status": "SUCCESS",
    "username": "admin",
    "stagePercent": 0
  }
]
```


Configuring Display of Inputs and Outputs

The entity Details page displays entity type-specific information, including table inputs and operation inputs and outputs. See Displaying Entity Details. In some cases, displaying inputs and outputs can delay rendering of the Details page. By default, displaying inputs and outputs is disabled. You can configure the Navigator Metadata Server to display inputs and outputs by setting the `nav.ui.details_io_enabled` property to `true` as follows:

1. Do one of the following:
   - Select **Clusters** > **Cloudera Management Service** > **Cloudera Management Service**.
   - On the **Home** > **Status** tab, 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.ui.details_io_enabled=true
```

   If more than one role group applies to this configuration, edit the value for the appropriate role group. See Modifying Configuration Properties Using Cloudera Manager on page 13.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

Managing Metadata Policies

To manage metadata policies, perform the following tasks:

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 **Home** > **Status** tab, 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: tcp://localhost:61616.
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 Using Cloudera Manager on page 13.

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 **Home** > **Status** tab, in **Cloudera Management Service** table, click the **Cloudera Management Service** link.

2. Click the **Configuration** tab.
3. Select **Category** > **Policies**.
4. Select or clear 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 Using Cloudera Manager on page 13.

5. Click **Save Changes** to commit the changes.
6. Restart the role.

How To Create a Multitenant Enterprise Data Hub

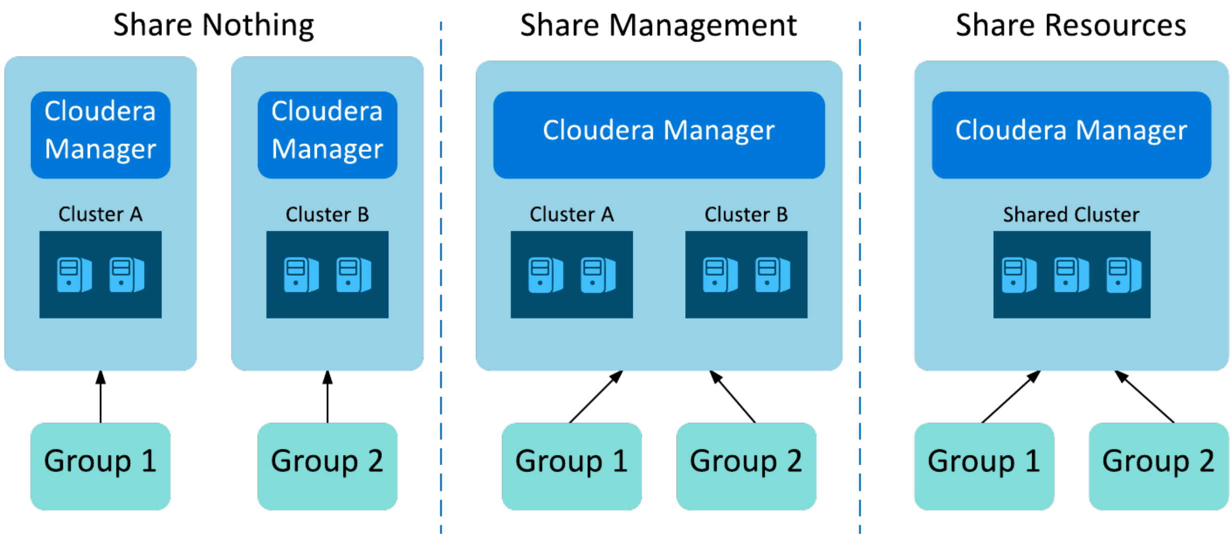
Multitenancy in an enterprise data hub (EDH) lets you share the collective resources of your CDH clusters between user groups without impacting application performance or compromising security.

Advantages of multitenancy include opportunities for data sharing, consolidated operations, improved performance, and better use of resources.

This topic walks through the steps to create a multitenant enterprise data hub:

Choosing an Isolation Model

There are three standard isolation models for an EDH: Share Nothing, Share Management, Share Resources.



Share Nothing

In a *share nothing* architecture, both management and data are completely separate. Nothing is shared between clusters. This architecture does not provide the benefits of multitenancy, but IT teams might find it appropriate based on specific operational realities and governance policies.

For example, specific company contracts or policies might force an enterprise IT team to use this model. Another common example is security and data privacy mandates that restrict the transfer of data sets across geographical boundaries.

Share Management

A *shared management* model offers the benefits of reduced operational overhead without sharing cluster resources or data between groups. This approach is a middle ground, granting some of the benefits of multitenancy while maintaining isolation at the cluster level. This is the preferred choice for environments where full multitenancy is not appropriate. For example, enterprises commonly employ this model for purpose-built, high-priority clusters that cannot risk any performance issues or resource contention, such as an ad serving system or retail personalization, “next offer” engine. While a multitenant EDH always faces the risks of misconfigured applications or malicious users, this model mitigates these risks at the cost of data sharing and resource pooling.

Share Data

The shared resource model uses full multitenancy with all the benefits from consolidated management to shared data and resources. It represents the desired end state for many EDH operators. For example, a biotechnology firm can harness the entire body and insight of research, trial data, and individual perspectives from all its research teams and other departments by employing a full multitenant EDH to store and analyze its information, greatly accelerating innovation through transparency and accessibility.

Balancing Criticality and Commonality

Enterprise IT teams often discover that multitenancy is not necessarily a good fit for their mission-critical workloads running uniquely tailored data sets. Multitenancy can be extremely useful for less critical workloads that employ shared data sets by reducing unnecessary or burdensome data duplication and synchronization. For most situations where business units share data, but the specific workloads are critical to the organization, the overarching business priorities and SLA goals drive the choice of a multitenant or isolated architecture. For some, the risk of latency and resource contention weighs heavily on their performance goals, and would suggest a shared management model. Others consider data visibility paramount, such as for fraud detection and insider threat analysis, which would warrant a shared resource model.

Configuring Security

Once you settle on an isolation model, you can choose the security elements to support your model.

Security for Hadoop is clearly critical in both single tenant and multitenant environments. It establishes the foundation for trusted data and usage among the various actors in the business environment. Without such trust, enterprises cannot rely on the resources and information when making business-critical decisions, which in turn undermines the benefits of operational consolidation and the decreased friction of shared data and insights. Cloudera's EDH provides a rich set of tools and frameworks for security. Key elements of this strategy and its systems include:

- **Authentication**, which proves users are who they say they are.
- **Authorization**, which determines what users are allowed to see and do.
- **Auditing**, which determines who did what, and when.
- **Data Protection**, which encrypts data-at-rest and in-motion.

Cloudera's EDH offers additional tools, such as network connectivity management and data masking. For further information on how IT teams can enable enterprise-grade security measures and policies for multitenant clusters, see Securing Your Enterprise Hadoop Ecosystem. In the context of multitenant administration, security requirements should also include:

- Delegating Security Management
- Managing Auditor Access
- Managing Data Visibility

Delegating Security Management

A central IT team tends to become the bottleneck in granting permissions to individuals and teams to specific data sets when handling large numbers of data sources with different access policies. Organizations can use Apache Sentry, the open source role-based access control (RBAC) system for Hadoop, to delegate permissions management for given data sets. Using this approach, local data administrators are responsible for assigning access for those data sets to the appropriate individuals and teams.

For more information, see Authorization With Apache Sentry.

Managing Auditor Access

For most large multitenant clusters, audit teams typically need access to data audit trails. For example, an audit team might need to monitor usage patterns for irregular behavior such as spikes in request access to credit card details or other sensitive information, without having full access to the cluster and its resources and data. Enterprise IT teams often adhere to the best practice of "least privilege" and restrict operational access to the minimum data and activity set required. For these cases, Cloudera Navigator provides a data auditor role that partitions the management rights to the cluster so that administrators can grant the audit team access only to the informational data needed, mitigating the impact to operations and security. This approach also answers the common request of audit teams to simplify and focus the application user interface.

For more information, see Cloudera Navigator Auditing.

Managing Data Visibility

Data visibility, in particular for the cluster administrator, is another security requirement that is prominent in most multitenant environments, especially those under strict compliance policies or regulations. Typical security approaches encrypt data, both on-disk and in-use, so that only users with the correct access can view data. Even administrators without proper access cannot view data stored on Hadoop. Cloudera Navigator provides data encryption and enterprise-grade key management with encrypt and key trustee, out-of-the-box.

For more information, see Cloudera Navigator Encryption.

Managing Resource Isolation

IT administrators must manage another crucial aspect of running a multitenant environment: facilitating the fair and equitable usage of finite cluster resources across different users and workloads. Typically, this is a result of aggregating resources to drive improved performance and utilization, a key business driver for multitenancy. Multiple groups within the organization finance the operations of this resource pool to meet this goal. As an outcome of many of these financing models, EDH administrators require systems to grant proportional access to this pool based on the proportion of payment. In addition, a successful multitenant environment employs these tools and frameworks to let users meet SLAs for critical workloads, even in the presence of unpredictable usage stemming from multiple, simultaneous workloads and ill-constructed or misconfigured processes.

Managing Resources

The practical batch processing engine of Hadoop, MapReduce, provides a scheduler framework that administrators can configure to ensure multiple, simultaneous user jobs share physical resources. More specifically, many production environments have successful implementations of the Fair Scheduler. These environments provide maximum utilization while enforcing SLAs through assigned resource minimums.

For more information, see Configuring the Fair Scheduler.

Defining Tenants with Dynamic Resource Pools

With the advent of computing capabilities such as Impala and Cloudera Search and a growing ecosystem of partner applications built to take advantage of CDH and the elements of Cloudera's EDH, cluster faculties and data are increasingly shared with systems that do not operate within the original management framework of MapReduce. A resource management solution must take the full range of these systems into account. To address this challenge, Cloudera's EDH and the underlying Hadoop platform, CDH, ship with the YARN resource management framework. In this context, YARN becomes a building block for computing engines to coordinate consumption and usage reservations to ensure resources are fairly allocated and used. This approach is sometimes referred to as *dynamic partitioning*.

Currently, Impala, MapReduce, and other well designed YARN applications participate in dynamic partitioning in CDH. IT administrators should also consider, with respect to the scheduler capability, how best to regulate tenant access to specified allocations (also known as *pools*) of resources. For example, IT teams might want to balance allocations between the processing needs for their marketing team's near real-time campaign dashboards and their finance department's SLA-driven quarterly compliance and reporting jobs. These administrative needs also extend to multiple applications within a single group. For example, the finance team must balance their quarterly reporting efforts with the daily expense report summaries. To achieve these goals, Hadoop and YARN support Access Control Lists for the various resource schedulers, ensuring that a user (or application) or group of users can only access a given resource pool.

For more information, see Dynamic Resource Pools.

Using Static Partitioning

While dynamic partitioning with YARN offers the IT administrator immense flexibility from a resource management perspective, IT teams operate applications that are not built on the YARN framework or require hard boundaries for resource allocation in order to separate them fully from other services in the cluster. Typically, these applications are purpose-built and by design do not permit this degree of resource flexibility.

To satisfy these business cases, Cloudera's EDH, through Cloudera Manager, supports a static partitioning model, which leverages a technology available on modern Linux operating systems called *container groups* (cgroups). In this model, IT administrators specify policies within the host operating system to restrict a particular service or application to a given allocation of cluster resources. For instance, the IT administrator can choose to partition a cluster by limiting an Apache HBase service to a maximum of 50% of the cluster resources and allotting the remaining 50% to a YARN service and its associated dynamic partitioning in order to accommodate the business SLAs and workloads handled by each of these services.

For more information, see Static Resource Pools.

Using Impala Admission Control

Within the constraints of the static service pool, you can further subdivide Impala's resources using Admission Control.

You use Admission Control to divide usage between Dynamic Resource Pools in multitenant use cases. Allocating resources judiciously allows your most important queries to run faster and more reliably.

For more information, see Managing Impala Admission Control.

Managing Quotas

Fair distribution of resources is essential for keeping tenants happy and productive.

While resource management systems ensure appropriate access and minimum resource amounts for running applications, IT administrators must also carefully govern cluster resources in terms of disk usage in a multitenant environment. Like resource management, disk management is a balance of business objectives and requirements across a range of user communities. The underlying storage foundation of a Hadoop-based EDH, the Hadoop Distributed File System (HDFS), supports quota mechanisms that administrators use to manage space usage by cluster tenants.

HDFS Utilization Reporting

Cloudera Manager reports let you keep track disk usage (storage space information) and directory usage (file metadata, such as size, owner, and last access date). You can use these reports to inform your decisions regarding quotas.

For more information, see Disk Usage Reports and Directory Usage Reports.

Managing Storage Quotas

Administrators can set disk space limits on a per-directory basis. This quota prevents users from accidentally or maliciously consuming too much disk space within the cluster, which can impact the operations of HDFS, similar to other file systems.

For more information, see Setting HDFS Quotas.

Managing Name Quotas

Name quotas are similar to disk quotas. Administrators use them to limit the number of files or subdirectories within a particular directory. This quota helps IT administrators optimize the metadata subsystem (NameNode) within the Hadoop cluster.

Name quotas and disk space quotas are the primary tools for administrators to make sure that tenants have access only to an appropriate portion of disk and name resources, much like the resource allocations mentioned in the previous section, and cannot adversely affect the operations of other tenants through their misuse of the shared file system.

For more information, see Setting HDFS Quotas.

Monitoring and Alerting

Resource and quota management controls are critical to smooth cluster operations. Even with these tools and systems, administrators have to plan for unforeseen situations such as an errant job or process that overwhelms an allotted resource partition for a single group and requires investigation and possible response.

Cloudera Manager provides Hadoop administrators a rich set of reporting and alerting tools that can be used to identify dangerous situations like low disk space conditions; once identified, Cloudera Manager can generate and send alerts to a network operations center (NOC) dashboard or an on-call resource via pager for immediate response.

For more information, see Introduction to Cloudera Manager Monitoring.

Implementing Showback and Chargeback

A common requirement for multitenant environments is the ability to meter the cluster usage of different tenants. As mentioned, one of the key business drivers of multitenancy is the aggregation of resources to improve utilization and performance. The multiple participants build internal budgets to finance this resource pool. In many organizations, IT departments use the metered information to drive showback or chargeback models and illustrate compliance.

Cluster Utilization Reporting

Cluster Utilization Report screens in Cloudera Manager display aggregated utilization information for YARN and Impala jobs. The reports display CPU utilization, memory utilization, resource allocations made due to the YARN fair scheduler, and Impala queries. The report displays aggregated utilization for the entire cluster and also breaks out utilization by tenant. You can configure the report to display utilization for a range of dates, specific days of the week, and time ranges.

Cluster utilization reporting lets you answer key questions such as:

- “How much CPU and memory did each tenant use?”
- “I set up fair scheduler. Did each of my tenants get their fair share?”
- “Which tenants had to wait the longest for their applications to get resources?
- “Which tenants asked for the most memory but used the least?”
- “When do I need to add nodes to my cluster?”

For more information, see Cluster Utilization Reports.

Appendix: Apache License, Version 2.0

SPDX short identifier: Apache-2.0

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License.

Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution.

You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

1. You must give any other recipients of the Work or Derivative Works a copy of this License; and
2. You must cause any modified files to carry prominent notices stating that You changed the files; and
3. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
4. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions.

Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks.

This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty.

Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability.

In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability.

While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work

To apply the Apache License to your work, attach the following boilerplate notice, with the fields enclosed by brackets "[]" replaced with your own identifying information. (Don't include the brackets!) The text should be enclosed in the appropriate comment syntax for the file format. We also recommend that a file or class name and description of purpose be included on the same "printed page" as the copyright notice for easier identification within third-party archives.

```
Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
```