cloudera
Ask Bigger Questions

CDH 5 Installation Guide

Cloudera, Inc.
1001 Page Mill Road Bldg 2
Palo Alto, CA 94304
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Release Information

Version: 5.1.x
Date: September 8, 2015

Table of Contents

About this Guide

This *CDH 5 Installation Guide* is for Apache Hadoop developers and system administrators interested in Hadoop installation. It describes how to install and configure version 5 of CDH, Cloudera's 100% open source platform containing Apache Hadoop and related projects (CDH 5), and how to deploy it on a cluster.

The guide covers the following major topics.

Before You Start:

- What's New in CDH 5 on page 21
- Before You Install CDH 5 on a Cluster on page 23

Installation tasks:

- Install CDH 5. Start here for a new installation on a cluster.
- Deploy CDH 5. Do these tasks after installing core Hadoop.
- Install components. Install additional components after installing and deploying HDFS and MapReduce. (Components are listed below.)

> **Note:** To install a release earlier than the current CDH 5 release (for example if you want to add new nodes to a cluster without upgrading the cluster to the latest release), follow these instructions.

Upgrade tasks:

- Upgrade from CDH 4 to CDH 5. Use these instructions if you are currently running a CDH 4 release; *or*
- Upgrade from a CDH 5 Beta Release. Use these instructions if you are currently running a CDH 5 Beta release; *or*
- Upgrade from CDH 5.0.0 or later. Use these instructions if you are currently running CDH 5.0.0 or later.
- Upgrade components. Upgrade all installed components (see the list below) after upgrading core Hadoop.

> **Note:** Use these instructions to migrate data from a CDH 4 cluster to a CDH 5 cluster.

CDH 5 Components

Use the following sections to install or upgrade CDH 5 components:

- Crunch Installation on page 155
- Flume Installation on page 157
- HBase Installation on page 169
- Installing and Using HCatalog on page 203
- Hive Installation on page 233
- HttpFS Installation on page 259
- Hue Installation on page 263
- Impala Installation on page 211
- Llama Installation on page 291
- Mahout Installation on page 293
- Oozie Installation on page 297
- Pig Installation on page 319
- Search Installation on page 325
- Sentry Installation on page 327
- Snappy Installation on page 329
- Spark Installation on page 333

- Sqoop 1 Installation on page 339
- Sqoop 2 Installation on page 345
- Whirr Installation on page 353
- ZooKeeper Installation

See also the instructions for installing or updating LZO.

Other Topics in this Guide

This guide also contains information on the following:

- Avro Usage on page 365
- Using the Parquet File Format with Impala, Hive, Pig, and MapReduce on page 371
- Configuring Ports for CDH 5
- Maintenance Tasks and Notes
- Java Development Kit (JDK) Installation
- Mountable HDFS
- Configuring an NFSv3 Gateway on page 393
- Creating a local Yum Repository
- Using the CDH 5 Maven Repository on page 401
- Building RPMs from Source RPMs
- Getting Support on page 405
- Apache and Third-Party Licenses

What's New in CDH 5

CDH 5 contains many changes and enhancements when compared to previous releases. For details about the changes in CDH 5, see the CDH 5 Release Notes, and specifically the section New Features in CDH 5. For links to the detailed change lists that describe the bug fixes and improvements to all of the CDH 5 projects, see the packaging section of CDH Version and Packaging Information.

Before You Install CDH 5 on a Cluster

> **Note: Running Services**
>
> When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

> **Important:**
>
> - **Upgrading from CDH 4:** If you are upgrading from CDH 4, you must first uninstall CDH 4, then install CDH 5; see Upgrading from CDH 4 to CDH 5.

Before you install CDH 5 on a cluster, there are some important steps you need to do to prepare your system:

1. Verify you are using a supported operating system for CDH 5. See CDH 5 Requirements and Supported Versions.
2. If you haven't already done so, install the Oracle Java Development Kit. For instructions and recommendations, see Java Development Kit Installation.

> **Important:**
>
> On SLES 11 platforms, do not install or try to use the IBM Java version bundled with the SLES distribution; Hadoop will not run correctly with that version. Install the Oracle JDK following directions under Java Development Kit Installation.

> **Note:**
>
> If you are migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN), see Migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN) on page 43 for important information and instructions.

High Availability

In CDH 5 you can configure high availability both for the NameNode and the JobTracker or Resource Manager.

- For more information and instructions on setting up a new HA configuration, see the CDH 5 High Availability Guide.

  > **Important:**
  >
  > If you decide to configure HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`. After completing the HDFS HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

- To upgrade an existing configuration, follow the instructions under Upgrading to CDH 5 on page 58.

CDH 5 and MapReduce

The default installation in CDH 5 is MapReduce 2.*x* (MRv2) built on the YARN framework. In this document we usually refer to this new version as **YARN**. CDH 5 also provides an implementation of the previous version of MapReduce, now referred to as **MRv1**. You can use the instructions on this page to install:

- YARN *or*
- MRv1 *or*
- both implementations.

Important:

MRv1 and YARN share a common set of configuration files, so it is safe to *configure* both of them so long as you *run* only one set of daemons at any one time. Cloudera does not support running MRv1 and YARN daemons on the same nodes at the same time; it will degrade performance and may result in an unstable cluster deployment.

Note:

If you are migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN), see Migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN) on page 43 for important information and instructions.

MapReduce 2.0 (YARN)

MapReduce has undergone a complete overhaul and CDH 5 now includes MapReduce 2 (MRv2). The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks. For details of the new architecture, see Apache Hadoop NextGen MapReduce (YARN).

See also Selecting Appropriate JAR files for your MRv1 and YARN Jobs on page 142.

CDH 5 Installation

This section describes how to install CDH 5. See the following sections for more information and instructions:

- CDH 5 and MapReduce on page 25
- Installing CDH 5
- Installing CDH 5 Components
- Apache Hadoop Documentation

Installing CDH 5

Ways To Install CDH 5

You can install CDH 5 in any of the following ways:

- Automated method using Cloudera Manager; instructions here. Cloudera Manager automates the installation and configuration of CDH 5 on an entire cluster if you have root or password-less sudo SSH access to your cluster's machines.

  Note: Cloudera recommends that you use the automated method if possible.

- Manual methods described below:
  - Download and install the CDH 5 1-click Install" package
  - Add the CDH 5 repository
  - Build your own CDH 5 repository

  If you use one of these methods rather than Cloudera Manager, the first of these methods (downloading and installing the "1-click Install" package) is recommended in most cases because it is simpler than building or adding a repository.

- Install from a CDH 5 tarball — see, the next topic, "How Packaging Affects CDH 5 Deployment".

How Packaging Affects CDH 5 Deployment

Installing from Packages

- To install and deploy YARN, follow the directions on this page and proceed with Deploying MapReduce v2 (YARN) on a Cluster.
- To install and deploy MRv1, follow the directions on this page and then proceed with Deploying MapReduce v1 (MRv1) on a Cluster.

Installing from a Tarball

Note: The instructions in this Installation Guide are tailored for a package installation, as described in the sections that follow, and do not cover installation or deployment from tarballs.

- If you install CDH 5 from a tarball, you will install YARN.
- In CDH 5, there is no separate tarball for MRv1. Instead, the MRv1 binaries, examples, etc., are delivered in the Hadoop tarball itself. The scripts for running MRv1 are in the `bin-mapreduce1` directory in the tarball, and the MRv1 examples are in the `examples-mapreduce1` directory.

Before You Begin Installing CDH 5 Manually

- The instructions on this page are for new installations. If you need to upgrade from an earlier release, see Upgrading from CDH 4 to CDH 5.
- For a list of supported operating systems, see CDH 5 Requirements and Supported Versions.
- These instructions assume that the sudo command is configured on the hosts where you will be doing the installation. If this is not the case, you will need the root user (superuser) to configure it.

Note:

If you are migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN), see Migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN) on page 43 for important information and instructions.

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Important:

- **Java Development Kit:** if you have not already done so, install the Oracle Java Development Kit (JDK); see Java Development Kit Installation.
- **Scheduler defaults:** note the following differences between MRv1 and MRv2 (YARN).
  - MRv1:
    - Cloudera Manager sets the default to FIFO.
    - CDH 5 sets the default to FIFO, with FIFO, Fair Scheduler, and Capacity Scheduler on the classpath by default.
  - MRv2 (YARN):
    - Cloudera Manager sets the default to Fair Scheduler.
    - CDH 5 sets the default to Fair Scheduler, with FIFO and Fair Scheduler on the classpath by default.
    - YARN does not support Capacity Scheduler.

High Availability

In CDH 5 you can configure high availability both for the NameNode and the JobTracker or Resource Manager.

- For more information and instructions on setting up a new HA configuration, see the CDH 5 High Availability Guide.

Important:

If you decide to configure HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`. After completing the HDFS HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

- To upgrade an existing configuration, follow the instructions under Upgrading to CDH 5 on page 58.

Steps to Install CDH 5 Manually

Step 1: Add or Build the CDH 5 Repository or Download the "1-click Install" package.

- If you are installing CDH 5 on a Red Hat system, you can download Cloudera packages using `yum` or your web browser.
- If you are installing CDH 5 on a SLES system, you can download the Cloudera packages using `zypper` or `YaST` or your web browser.
- If you are installing CDH 5 on an Ubuntu or Debian system, you can download the Cloudera packages using `apt` or your web browser.

On Red Hat-compatible Systems

Use one of the following methods to add or build the CDH 5 repository or download the package on Red Hat-compatible systems.

Note:

Use only one of the three methods.

- Download and install the CDH 5 "1-click Install" package *OR*
- Add the CDH 5 repository *OR*
- Build a Yum Repository

Do this on all the systems in the cluster.

To download and install the CDH 5 "1-click Install" package:

1. Click the entry in the table below that matches your Red Hat or CentOS system, choose **Save File**, and save the file to a directory to which you have write access (it can be your home directory).

OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS/Oracle 6	Red Hat/CentOS/Oracle 6 link

2. Install the RPM. For Red Hat/CentOS/Oracle 5:

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
```

For Red Hat/CentOS/Oracle 6 (64-bit):

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
```

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

OR: **To add the CDH 5 repository:**

Click the entry in the table below that matches your Red Hat or CentOS system, navigate to the repo file for your system and save it in the `/etc/yum.repos.d/` directory.

For OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS/Oracle 6 (64-bit)	Red Hat/CentOS/Oracle 6 link

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

***OR:* To build a Yum repository:**

If you want to create your own `yum` repository, download the appropriate repo file, create the repo, distribute the repo file and set up a web server, as described under Creating a Local Yum Repository.

Now continue with Step 2: Optionally Add a Repository Key on page 32, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo yum clean all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

On SLES Systems

Use one of the following methods to download the CDH 5 repository or package on SLES systems.

Note:

Use only one of the three methods.

- Download and install the CDH 5 "1-click Install" Package*OR*
- Add the CDH 5 repository*OR*
- Build a SLES Repository

To download and install the CDH 5 "1-click Install" package:

1. Download the CDH 5 "1-click Install" package.

   Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).

2. Install the RPM:

```
$ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
```

3. Update your system package index by running:

```
$ sudo zypper refresh
```

Now continue with Step 2: Optionally Add a Repository Key on page 32, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

***OR:* To add the CDH 5 repository:**

1. Run the following command:

```
$ sudo zypper addrepo -f
http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/cloudera-cdh5.repo
```

2. Update your system package index by running:

```
$ sudo zypper refresh
```

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

OR: **To build a SLES repository:**

If you want to create your own SLES repository, create a mirror of the CDH SLES directory by following these instructions that explain how to create a SLES repository from the mirror.

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo zypper clean --all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

On Ubuntu or Debian Systems

Use one of the following methods to download the CDH 5 repository or package.

Note:

Use only one of the three methods.

- Download and install the CDH 5 "1-click Install" Package *OR*
- Add the CDH 5 repository*OR*
- Build a Debian Repository

To download and install the CDH 5 "1-click Install" package:

1. Download the CDH 5 "1-click Install" package:

OS Version	Click this Link
Wheezy	Wheezy link
Precise	Precise link

2. Install the package. Do one of the following:
   - Choose **Open with** in the download window to use the package manager.
   - Choose **Save File**, save the package to a directory to which you have write access (it can be your home directory) and install it from the command line, for example:

```
sudo dpkg -i cdh5-repository_1.0_all.deb
```

Now continue with Step 2: Optionally Add a Repository Key on page 32, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

***OR:* To add the CDH 5 repository:**

Create a new file `/etc/apt/sources.list.d/cloudera.list` with the following contents:

- For Ubuntu systems:

```
deb [arch=amd64] http://archive.cloudera.com/cdh5/<OS-release-arch><RELEASE>-cdh5
 contrib
deb-src http://archive.cloudera.com/cdh5/<OS-release-arch><RELEASE>-cdh5 contrib
```

- For Debian systems:

```
deb http://archive.cloudera.com/cdh5/<OS-release-arch><RELEASE>-cdh5 contrib
deb-src http://archive.cloudera.com/cdh5/<OS-release-arch><RELEASE>-cdh5 contrib
```

where: <OS-release-arch> is `debian/wheezy/amd64/cdh` or `ubuntu/precise/amd64/cdh`, and <RELEASE> is the name of your distribution, which you can find by running `lsb_release -c`.

For example, to install CDH 5 for 64-bit Ubuntu Precise:

```
deb [arch=amd64] http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh precise-cdh5
 contrib
deb-src http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh precise-cdh5 contrib
```

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

***OR:* To build a Debian repository:**

If you want to create your own `apt` repository, create a mirror of the CDH Debian directory and then create an apt repository from the mirror.

Now continue with Step 1a: Optionally Add a Repository Key, and then choose Step 3: Install CDH 5 with YARN on page 33, or Step 4: Install CDH 5 with MRv1 on page 35; or do both steps if you want to install both implementations.

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo apt-get update
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

Step 2: Optionally Add a Repository Key

Before installing YARN or MRv1: (Optionally) add a repository key on each system in the cluster. Add the Cloudera Public GPG Key to your repository by executing one of the following commands:

- **For Red Hat/CentOS/Oracle 5 systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Red Hat/CentOS/Oracle 6 systems:**

  ```
  $ sudo rpm --import
  http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
  ```

- **For all SLES systems:**

  ```
  $ sudo rpm --import
  http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
  ```

- **For Ubuntu Precise systems:**

  ```
  $ curl -s http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh/archive.key
  | sudo apt-key add -
  ```

- **For Debian Wheezy systems:**

  ```
  $ curl -s http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh/archive.key
  | sudo apt-key add -
  ```

This key enables you to verify that you are downloading genuine packages.

Step 3: Install CDH 5 with YARN

Note:

Skip this step if you intend to use *only* MRv1. Directions for installing MRv1 are in Step 3.

To install CDH 5 with YARN:

Note:

If you decide to configure HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`. After completing the HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

1. Install and deploy ZooKeeper.

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
Resource Manager host (analogous to MRv1 JobTracker) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager`

Where to install	Install commands
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager`
NameNode host running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the Resource Manager running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
Ubuntu or Debian	`sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
One host in the cluster running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
Ubuntu or Debian	`sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
All client hosts running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-client`

Where to install	Install commands
SLES	`sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get install hadoop-client`

Note:

The `hadoop-yarn` and `hadoop-hdfs` packages are installed on each system automatically as dependencies of the other packages.

Step 4: Install CDH 5 with MRv1

Note:

If you are also installing YARN, you can skip any packages you have already installed in Step 3: Install CDH 5 with YARN on page 33.

Skip this step and go to Step 3: Install CDH 5 with YARN on page 33 if you intend to use *only* YARN.

Important:

Before proceeding, you need to decide:

1. Whether to configure High Availability (HA) for the NameNode and/or JobTracker; see the CDH 5 High Availability Guide for more information and instructions.
2. Where to deploy the NameNode, Secondary NameNode, and JobTracker daemons. As a general rule:
   - The NameNode and JobTracker run on the same "master" host unless the cluster is large (more than a few tens of nodes), and the master host (or hosts) should not run the Secondary NameNode (if used), DataNode or TaskTracker services.
   - In a large cluster, it is especially important that the Secondary NameNode (if used) runs on a separate machine from the NameNode.
   - Each node in the cluster **except the master host(s)** should run the DataNode and TaskTracker services.

If you decide to configure HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`. After completing the HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

1. Install and deploy ZooKeeper.

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
JobTracker host running:	

Where to install	Install commands
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-jobtracker`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-jobtracker`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-jobtracker`
NameNode host running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the JobTracker, NameNode, and Secondary (or Standby) NameNode hosts running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
Ubuntu or Debian	`sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
All client hosts running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-client`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get install hadoop-client`

Step 5: (Optional) Install LZO

If you decide to install LZO (Lempel–Ziv–Oberhumer compression), proceed as follows.

> **Note:**
>
> If you are upgrading to a new version of LZO, rather than installing it for the first time, you must first remove the old version; for example, on a RHEL system:
>
> ```
> yum remove hadoop-lzo
> ```

1. Add the repository on each host in the cluster. Follow the instructions for your OS version:

For OS Version	Do this
Red Hat/CentOS/Oracle 5	Navigate to this link and save the file in the `/etc/yum.repos.d/` directory.
Red Hat/CentOS 6	Navigate to this link and save the file in the `/etc/yum.repos.d/` directory.
SLES	1. Run the following command: `$ sudo zypper addrepo -f http://archive.cloudera.com/gplextras5/sles/11/x86_64/gplextras/cloudera-gplextras5.repo` 2. Update your system package index by running: `$ sudo zypper refresh`
Ubuntu or Debian	Navigate to this link and save the file as `/etc/apt/sources.list.d/gplextras.list`. **Important:** Make sure you do not let the file name default to `cloudera.list`, as that will overwrite your existing `cloudera.list`.

2. Install the package on each host as follows:

For OS version	Install commands
Red Hat/CentOS compatible	`sudo yum install hadoop-lzo`
SLES	`sudo zypper install hadoop-lzo`
Ubuntu or Debian	`sudo apt-get install hadoop-lzo`

3. Continue with installing and deploying CDH. As part of the deployment, you will need to do some additional configuration for LZO, as shown under Configuring LZO on page 128.

> **Important:** Make sure you do this configuration *after* you have copied the default configuration files to a custom location and set alternatives to point to it.

Step 6: Deploy CDH and Install Components

Now proceed with:

- deploying CDH 5
- installing components.

Installing CDH 5 Components

In a new installation, you should install and deploy CDH before proceeding to install the components listed below. See CDH 5 Installation on page 27 and Deploying CDH 5 on a Cluster on page 117.

CDH 5 Components

Use the following sections to install or upgrade CDH 5 components:

- Crunch Installation on page 155
- Flume Installation on page 157
- HBase Installation on page 169
- Installing and Using HCatalog on page 203
- Hive Installation on page 233
- HttpFS Installation on page 259
- Hue Installation on page 263
- Impala Installation on page 211
- Llama Installation on page 291
- Mahout Installation on page 293
- Oozie Installation on page 297
- Pig Installation on page 319
- Search Installation on page 325
- Sentry Installation on page 327
- Snappy Installation on page 329
- Spark Installation on page 333
- Sqoop 1 Installation on page 339
- Sqoop 2 Installation on page 345
- Whirr Installation on page 353
- ZooKeeper Installation

See also the instructions for installing or updating LZO.

Viewing the Apache Hadoop Documentation

- For additional Apache Hadoop documentation, see http://archive.cloudera.com/cdh5/cdh/5/hadoop.
- For more information about YARN, see the Apache Hadoop NextGen MapReduce (YARN) page at http://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-yarn/hadoop-yarn-site/YARN.html.

Installing an Earlier CDH 5 Release

Follow these instructions to install a CDH 5 release that is **earlier than the current CDH 5 release**.

A common reason for doing this would be that you need to add new nodes to an existing cluster that is not running the most recent version of CDH 5. For example your cluster might be running CDH 5.0.1 when the most recent release is CDH 5.1.0; in this case, you will want to install CDH 5.0.1 on the new nodes, not CDH 5.1.0.

- **Warning:**

  Do not attempt to use these instructions to roll your cluster back to a previous release. Use them only to expand an existing cluster that you do not want to upgrade, or to create a new cluster running a version of CDH 5 that is earlier than the current CDH 5 release.

Downloading and Installing an Earlier Release

Choose your Linux version and proceed as follows to install an earlier release:

- On Red Hat-compatible systems
- On SLES systems
- On Ubuntu and Debian systems

On Red Hat-compatible systems

Step 1. Download and save the Yum repo file

Click the entry in the table below that matches your Red Hat or CentOS system, navigate to the repo file for your system and save it in the `/etc/yum.repos.d/` directory.

For OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS 6 (64-bit)	Red Hat/CentOS 6 link

Step 2. Edit the repo file

Open the repo file you have just saved and change the `5` at the end of the line that begins `baseurl=` to the version number you want.

For example, if you have saved the file for Red Hat 6, it will look like this when you open it for editing:

```
[cloudera-cdh5]
name=Cloudera's Distribution for Hadoop, Version 5
baseurl=http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/
gpgkey = http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

If you want to install CDH 5.0.1, for example, change
`baseurl=http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/` to

`baseurl=http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5.0.1/`

In this example, the resulting file should look like this:

```
[cloudera-cdh5]
name=Cloudera's Distribution for Hadoop, Version 5
baseurl=http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5.0.1/
gpgkey = http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

Step 3: Proceed with the installation

1. Go to http://www.cloudera.com/content/support/en/documentation/cdh5-documentation/cdh5-documentation-v5-latest.html.
2. Use the `Select a Product Version` scroller to find the release you want, for example `CDH 5.0.x`
3. Find the CDH Installation Guide for your release.
4. Follow the instructions for Red Hat on the "Installing CDH 5" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH 5 with MRv1 or YARN, and is usually Step 2.)

On SLES systems

Step 1. Add the Cloudera repo

1. Run the following command:

```
$ sudo zypper addrepo -f
http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/cloudera-cdh5.repo
```

2. Update your system package index by running:

```
$ sudo zypper refresh
```

Step 2. Edit the repo file

Open the repo file that you have just added to your system and change the 4 at the end of the line that begins `baseurl=` to the version number you want.

The file should look like this when you open it for editing:

```
[cloudera-cdh4]
name=Cloudera's Distribution for Hadoop, Version 5
baseurl=http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/5/
gpgkey = http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

If you want to install CDH5.0.1, for example, change
`baseurl=http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/5/` to

`baseurl= http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/5.0.1/`

In this example, the resulting file should look like this:

```
[cloudera-cdh4]
name=Cloudera's Distribution for Hadoop, Version 5
baseurl=http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/5.0.1/
gpgkey = http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

Step 3: Proceed with the installation

1. Go to http://www.cloudera.com/content/support/en/documentation/cdh5-documentation/cdh5-documentation-v5-latest.html.
2. Use the `Select a Product Version` scroller to find the release you want, for example `CDH 5.0.x`
3. Find the CDH Installation Guide for your release.
4. Follow the instructions for SLES on the "Installing CDH 5" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH 5 with MRv1 or YARN, and is usually Step 2.)

On Ubuntu and Debian systems

Proceed as follows to add the Cloudera repo for your operating-system version and the Cloudera release you need.

Step 1: Create the repo File

Create a new file `/etc/apt/sources.list.d/cloudera.list` with the following contents:

- For Ubuntu systems:

```
deb [arch=amd64] http://archive.cloudera.com/cdh5/ <OS-release-arch> <RELEASE>-cdh5
 contrib deb-src http://archive.cloudera.com/cdh5/ <OS-release-arch> <RELEASE>-cdh5
 contrib
```

- For Debian systems:

```
deb http://archive.cloudera.com/cdh5/ <OS-release-arch> <RELEASE>-cdh5 contrib
deb-src http://archive.cloudera.com/cdh5/ <OS-release-arch> <RELEASE>-cdh5 contrib
```

where: <OS-release-arch> is `debian/wheezy/amd64/cdh` or `ubuntu/precise/amd64/cdh`, and <RELEASE> is the name of your distribution, which you can find by running `lsb_release -c`.

Now replace `-cdh5` near the end of each line (before `contrib`) with the CDH release you need to install. Here are some examples using CDH5.0.1:

For 64-bit Ubuntu Precise:

```
deb [arch=amd64] http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh
precise-cdh5.0.1 contrib
deb-src http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh precise-cdh5.0.1
contrib
```

For Debian Wheezy:

```
deb http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh wheezy-cdh5.0.1 contrib
deb-src http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh wheezy-cdh5.0.1 contrib
```

Step 2: Proceed with the installation

1. Go to http://www.cloudera.com/content/support/en/documentation/cdh5-documentation/cdh5-documentation-v5-latest.html.
2. Use the `Select a Product Version` scroller to find the release you want, for example `CDH 5.0.x`
3. Find the CDH Installation Guide for your release.
4. Follow the instructions for Ubuntu or Debian on the "Installing CDH 5" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH5 with MRv1 or YARN, and is usually Step 2.)

Migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN)

This is a guide to migrating from Apache MapReduce 1 (MRv1) to the Next Generation MapReduce (MRv2 or YARN).

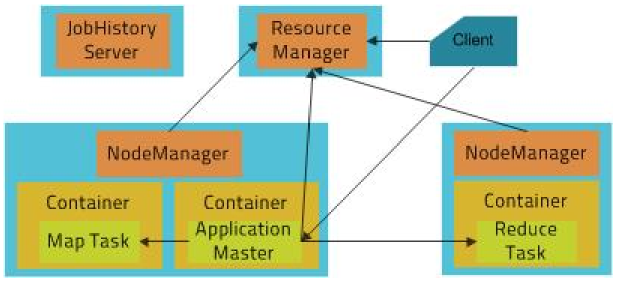
Introduction

MapReduce 2, or Next Generation MapReduce, is a long needed upgrade to the way that scheduling, resource management, and execution occur in Hadoop. At their core, the improvements separate cluster resource management capabilities from MapReduce-specific logic. They enable Hadoop to share resources dynamically between MapReduce and other parallel processing frameworks, such as Impala, allow more sensible and finer-grained resource configuration for better cluster utilization, and permit it to scale to accommodate more and larger jobs.

This document provides a guide to both the architectural and user-facing changes, so that both cluster operators and MapReduce programmers can easily make the transition.

Terminology and Architecture

MapReduce from Hadoop 1 (MapReduce 1) has been split into two components. The cluster resource management capabilities have become YARN (Yet Another Resource Negotiator), while the MapReduce-specific capabilities remain MapReduce. In the MapReduce 1 architecture, the cluster was managed by a service called the JobTracker. TaskTracker services lived on each node and would launch tasks on behalf of jobs. The JobTracker would serve information about completed jobs. In MapReduce 2, the functions of the JobTracker have been split between three services. The ResourceManager is a persistent YARN service that receives and runs applications (a MapReduce job is an application) on the cluster. It contains the scheduler, which, as previously, is pluggable. The MapReduce-specific capabilities of the JobTracker have been moved into the MapReduce Application Master, one of which is started to manage each MapReduce job and terminated when the job completes. The JobTracker's function of serving information about completed jobs has been moved to the JobHistoryServer. The TaskTracker has been replaced with the NodeManager, a YARN service that manages resources and deployment on a node. It is responsible for launching containers, each of which can house a map or reduce task.



The new architecture has its advantages. First, by breaking up the JobTracker into a few different services, it avoids many of the scaling issues faced by MapReduce in Hadoop 1. More importantly, it makes it possible to run frameworks other than MapReduce on a Hadoop cluster. For example, Impala can also run on YARN and share resources on a cluster with MapReduce.

For MapReduce Programmers: Writing and Running Jobs

Nearly all jobs written for CDH 4 MRv1 will be able to run without any modifications on an MRv2 cluster.

Java API Compatibility from CDH 4

MRv2 supports both the old ("mapred") and new ("mapreduce") MapReduce APIs used for MRv1, with a few caveats. The difference between the old and new APIs, which concerns user-facing changes, should not be confused with the difference between MRv1 and MRv2, which concerns changes to the underlying framework. CDH 4 and CDH 5 both support the new and old MapReduce APIs.

In general, applications that use @Public/@Stable APIs will be binary-compatible from CDH 4, meaning that compiled binaries should be able to run without modifications on the new framework. Source compatibility may be broken for applications that make use of a few obscure APIs that are technically public, but rarely needed and primarily exist for internal use. These APIs are detailed below. Source incompatibility means that code changes will be required to compile. It is orthogonal to binary compatibility - binaries for an application that is binary-compatible, but not source-compatible, will continue to run fine on the new framework, but code changes will be required to regenerate those binaries.

	Binary Incompatibilities	Source Incompatibilities
CDH 4 MRv1 to CDH 5 MRv1	None	None
CDH 4 MRv1 to CDH 5 MRv2	None	Rare
CDH 5 MRv1 to CDH 5 MRv2	None	Rare

The following are the known source incompatibilities:

- `KeyValueLineRecordReader#getProgress` and `LineRecordReader#getProgress` now throw IOExceptions in both the old and new APIs. Their superclass method, `RecordReader#getProgress`, already did this, but source compatibility will be broken for the rare code that used it without a try/catch block.
- `FileOutputCommitter#abortTask` now throws an IOException. Its superclass method always did this, but source compatibility will be broken for the rare code that used it without a try/catch block. This was fixed in CDH 4.3 MRv1 to be compatible with MRv2.
- `Job#getDependentJobs`, an API marked @Evolving, now returns a List instead of an ArrayList.

Compiling Jobs Against MRv2

If you are using Maven, compiling against MRv2 requires including the same artifact, `hadoop-client`. Changing the version to Hadoop 2 version (for example, using 2.2.0-cdh5.0.0 instead of 2.0.0-mr1-cdh4.3.0) should be enough. If you are not using Maven, compiling against all the Hadoop jars is recommended. A comprehensive list of Hadoop Maven artifacts is available at: Using the CDH 5 Maven Repository.

Job Configuration

As in MRv1, job configuration options can be specified on the command line, in Java code, or in the `mapred-site.xml` on the client machine in the same way they previously were. The vast majority of job configuration options that were available in MRv1 work in MRv2 as well. For consistency and clarity, many options have been given new names. The older names are deprecated, but will still work for the time being. The exceptions to this are `mapred.child.ulimit` and all options relating to JVM reuse, as these are no longer supported.

Submitting and Monitoring Jobs

The MapReduce command line interface remains entirely compatible. Use of the Hadoop command line tool to run MapReduce related commands (`pipes, job, queue, classpath, historyserver, distcp, archive`) is deprecated, but still works. The `mapred` command line tool is preferred for these commands.

Requesting Resources

A MapReduce job submission includes the amount of resources to reserve for each map and reduce task. As in MapReduce 1, the amount of memory requested is controlled by the `mapreduce.map.memory.mb` and `mapreduce.reduce.memory.mb` properties.

MapReduce 2 adds additional parameters that control how much processing power to reserve for each task as well. The `mapreduce.map.cpu.vcores` and `mapreduce.reduce.cpu.vcores` properties express how much parallelism a map or reduce task can take advantage of. These should remain at their default value of 1 unless your code is explicitly spawning extra compute-intensive threads.

For Administrators: Configuring and Running MRv2 Clusters

Configuration Migration

Since MapReduce 1 functionality has been split into two components, MapReduce cluster configuration options have been split into YARN configuration options, which go in `yarn-site.xml`, and MapReduce configuration options, which go in `mapred-site.xml`. Many have been given new names to reflect the shift. As JobTrackers and TaskTrackers no longer exist in MRv2, all configuration options pertaining to them no longer exist, although many have corresponding options for the ResourceManager, NodeManager and JobHistoryServer.

A minimal configuration required to run MRv2 jobs on YARN is:

- `yarn-site.xml` configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>yarn.resourcemanager.hostname</name>
    <value>you.hostname.com</value>
  </property>

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
</configuration>
```

- `mapred-site.xml` configuration

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
</configuration>
```

See Deploying MapReduce v2 (YARN) on a Cluster on page 135 for instructions for a full deployment.

Resource Configuration

One of the larger changes in MRv2 is the way that resources are managed. In MRv1, each node was configured with a fixed number of map slots and a fixed number of reduce slots. Under YARN, there is no distinction between resources available for maps and resources available for reduces - all resources are available for both. Second, the notion of slots has been discarded, and resources are now configured in terms of amounts of memory (in megabytes) and CPU (in “virtual cores”, which are described below). Resource configuration is an inherently difficult topic, and the added flexibility that YARN provides in this regard also comes with added complexity. Cloudera Manager will pick sensible values automatically, but if you are setting up your cluster manually or just interested in the details, read on.

Resource Requests

From the perspective of a developer requesting resource allocations for a job's tasks, nothing needs to be changed. Map and reduce task memory requests still work and, additionally, tasks that will use multiple threads can request more than 1 core with the `mapreduce.map.cpu.vcores` and `mapreduce.reduce.cpu.vcores` `properties`.

Configuring Node Capacities

In MRv1, the `mapred.tasktracker.map.tasks.maximum` and `mapred.tasktracker.reduce.tasks.maximum` properties dictated how many map and reduce slots each TaskTracker had. These properties no longer exist in YARN. Instead, YARN uses `yarn.nodemanager.resource.memory-mb` and `yarn.nodemanager.resource.cpu-vcores`, which control the amount of memory and CPU on each node, both available to both maps and reduces. If you were using Cloudera Manager to configure these automatically, Cloudera Manager will take care of it in MRv2 as well. If configuring these manually, simply set these to the amount of memory and number of cores on the machine after subtracting out resources needed for other services.

Virtual Cores

To better handle varying CPU requests, YARN supports virtual cores (vcores) , a resource meant to express parallelism. The "virtual" in the name is somewhat misleading - on the NodeManager, vcores should be configured equal to the number of physical cores on the machine. Tasks should be requested with vcores equal to the number of cores they can saturate at once. Currently vcores are very coarse - tasks will rarely want to ask for more than one of them, but a complementary axis that represents processing power may be added in the future to enable finer-grained resource configuration.

Rounding Request Sizes

Also noteworthy are the `yarn.scheduler.minimum-allocation-mb`, `yarn.scheduler.minimum-allocation-vcores`, `yarn.scheduler.increment-allocation-mb`, and `yarn.scheduler.increment-allocation-vcores` properties, which default to 1024, 1, 512, and 1 respectively. If tasks are submitted with resource requests lower than the minimum-allocation values, their requests will be set to these values. If tasks are submitted with resource requests that are not multiples of the increment-allocation values, their requests will be rounded up to the nearest increments.

To make all of this more concrete, let's use an example. Each node in the cluster has 24 GB of memory and 6 cores. Other services running on the nodes require 4 GB and 1 core, so we set `yarn.nodemanager.resource.memory-mb` to 20480 and `yarn.nodemanager.resource.cpu-vcores` to 5. If you leave the map and reduce task defaults of 1024 MB and 1 virtual core intact, you will have at most 5 tasks running at the same time. If you want each of your tasks to use 5 GB, set their `mapreduce.(map|reduce).memory.mb` to 5120, which would limit you to 4 tasks running at the same time.

Scheduler Configuration

Cloudera supports use of the Fair and FIFO schedulers in MRv2. Fair Scheduler allocation files require changes in light of the new way that resources work. The `minMaps, maxMaps, minReduces,` and `maxReduces` queue properties have been replaced with a `minResources` property and a `maxProperties`. Instead of taking a number of slots, these properties take a value like "1024 MB, 3 vcores". By default, the MRv2 Fair Scheduler will attempt to equalize memory allocations in the same way it attempted to equalize slot allocations in MRv1. The MRv2 Fair Scheduler contains a number of new features including hierarchical queues and fairness based on multiple resources.

- **Important:**

  Cloudera does not support the Capacity Scheduler in YARN.

Administration Commands

The `jobtracker` and `tasktracker` commands, which start the JobTracker and TaskTracker, are no longer supported because these services no longer exist. They are replaced with "`yarn resourcemanager`" and "`yarn nodemanager`", which start the ResourceManager and NodeManager respectively. "`hadoop mradmin`" is no longer supported. Instead, "`yarn rmadmin`" should be used. The new admin commands mimic the functionality of the MRv1 names, allowing nodes, queues, and ACLs to be refreshed while the ResourceManager is running.

Security

The following section outlines the additional changes needed to migrate a secure cluster.

New YARN Kerberos service principals should be created for the ResourceManager and NodeManager, using the pattern used for other Hadoop services, i.e. yarn@<HOST>. The mapred principal should still be used for the JobHistoryServer. If you are using Cloudera Manager to configure security, this will be taken care of automatically.

As in MRv1, a configuration must be set to have the user that submits a job own its task processes. The equivalent of MRv1's LinuxTaskController is the LinuxContainerExecutor. In a secure setup, NodeManager configurations should set `yarn.nodemanager.container-executor.class` to `org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor`. Properties set in the `taskcontroller.cfg` configuration file should be migrated to their analagous properties in the `container-executor.cfg` file.

In secure setups, configuring `hadoop-policy.xml` allows administrators to set up access control lists on internal protocols. The following is a table of MRv1 options and their MRv2 equivalents:

MRv1	MRv2	Comment
`security.task.umbilical.protocol.acl`	`security.job.task.protocol.acl`	As in MRv1, this should never be set to anything other than *
`security.inter.tracker.protocol.acl`	`security.resourcetracker.protocol.acl`	
`security.job.submission.protocol.acl`	`security.applicationclient.protocol.acl`	
`security.admin.operations.protocol.acl`	`security.resourcemanager-administration.protocol.acl`	
	`security.applicationmaster.protocol.acl`	No MRv1 equivalent
	`security.containermanagement.protocol.acl`	No MRv1 equivalent
	`security.resourcelocalizer.protocol.acl`	No MRv1 equivalent
	`security.job.client.protocol.acl`	No MRv1 equivalent

Queue access control lists (ACLs) are now placed in the Fair Scheduler configuration file instead of the JobTracker configuration. A list of users and groups that can submit jobs to a queue can be placed in `aclSubmitApps` in the queue's configuration. The queue administration ACL is no longer supported, but will be in a future release.

Ports

The following is a list of default ports used by MRv2 and YARN, as well as the configuration properties used to configure them.

Port	Use	Property
8032	ResourceManager Client RPC	`yarn.resourcemanager.address`
8030	ResourceManager Scheduler RPC (for ApplicationMasters)	`yarn.resourcemanager.scheduler.address`

Port	Use	Property
8033	ResourceManager Admin RPC	`yarn.resourcemanager.admin.address`
8088	ResourceManager Web UI and REST APIs	`yarn.resourcemanager.webapp.address`
8031	ResourceManager Resource Tracker RPC (for NodeManagers)	yarn.resourcemanager.resource-tracker.address
8040	NodeManager Localizer RPC	`yarn.nodemanager.localizer.address`
8042	NodeManager Web UI and REST APIs	`yarn.nodemanager.webapp.address`
10020	Job History RPC	`mapreduce.jobhistory.address`
19888	Job History Web UI and REST APIs	`mapreduce.jobhistory.webapp.address`
13562	Shuffle HTTP	`mapreduce.shuffle.port`

High Availability

YARN supports ResourceManager HA to make a YARN cluster highly-available; the underlying architecture of Active / Standby pair is similar to JobTracker HA in MRv1. A major improvement over MRv1 is: in YARN, the completed tasks of in-flight MapReduce jobs are not re-run on recovery after the ResourceManager is restarted or failed over. Further, the configuration and setup has also been simplified. The main differences are:

1. Failover controller has been moved from a separate ZKFC daemon to be a part of the ResourceManager itself. So, there is no need to run an additional daemon.
2. Clients, Applications, and NodeManagers do not require configuring a proxy-provider to talk to the active ResourceManager.

Below is a table with HA-related configurations used in MRv1 and their equivalents in YARN:

MRv1	YARN / MRv2	Comment
`mapred.jobtrackers.<name>`	`yarn.resourcemanager.ha.rm-ids`	
`mapred.ha.jobtracker.id`	`yarn.resourcemanager.ha.id`	Unlike in MRv1, this must be configured in YARN.
`mapred.jobtracker.<rpc-address>.<name>.<id>`	yarn.resourcemanager.<rpc-address>.<id>	YARN/ MRv2 has different RPC ports for different functionalities. Each port-related configuration must be suffixed with an id. Note that there is no <name> in YARN.
`mapred.ha.jobtracker.rpc-address.<name>.<id>`	yarn.resourcemanager.ha.admin.address	
`mapred.ha.fencing.methods`	`yarn.resourcemanager.ha.fencer`	Not required to be specified
`mapred.client.failover.*`	None	Not required
	`yarn.resourcemanager.ha.enabled`	Enable HA
`mapred.jobtracker.restart.recover`	yarn.resourcemanager.recovery.enabled	Enable recovery of jobs after failover

MRv1	YARN / MRv2	Comment
	yarn.resourcemanager.store.class	org.apache .hadoop.yarn .server.resourcemanager .recovery .ZKRMStateStore
mapred.ha.automatic-failover.enabled	yarn.resourcemanager.ha.auto-failover.enabled	Enable automatic failover
mapred.ha.zkfc.port	yarn.resourcemanager.ha.auto-failover.port	
mapred.job.tracker	yarn.resourcemanager.cluster.id	Cluster name

Upgrading an MRv1 Installation with Cloudera Manager

See Importing MapReduce Configurations to YARN for instructions.

Manually Upgrading an MRv1 Installation

The following packages are no longer used in MRv2 and should be uninstalled: `hadoop-0.20-mapreduce`, `hadoop-0.20-mapreduce-jobtracker`, `hadoop-0.20-mapreduce-tasktracker`, `hadoop-0.20-mapreduce-zkfc`, `hadoop-0.20-mapreduce-jobtrackerha`

The following additional packages must be installed: `hadoop-yarn`, `hadoop-mapreduce`, `hadoop-mapreduce-historyserver`, `hadoop-yarn-resourcemanager`, `hadoop-yarn-nodemanager`.

The next step is to look at all the service configurations placed in `mapred-site.xml` and replace them with their corresponding YARN configuration. Configurations starting with "yarn" should be placed inside `yarn-site.xml`, not `mapred-site.xml`. Refer to the Resource Configuration section above for best practices on how to convert TaskTracker slot capacities (`mapred.tasktracker.map.tasks.maximum` and `mapred.tasktracker.reduce.tasks.maximum`) to NodeManager resource capacities (`yarn.nodemanager.resource.memory-mb` and `yarn.nodemanager.resource.cpu-vcores`), as well as how to convert configurations in the Fair Scheduler allocations file, `fair-scheduler.xml`.

Finally, you can start the ResourceManager, NodeManagers and the JobHistoryServer.

Web UI

In MapReduce 1, the JobTracker Web UI served detailed information about the state of the cluster and the jobs (recent and current) running on it. It also contained the job history page, which served information from disk about older jobs.

The MapReduce 2 Web UI provides the same information structured in the same way, but has been revamped with a new look and feel. The ResourceManager's UI, which includes information about running applications and the state of the cluster, is now located by default at <ResourceManager host>:8088. The JobHistory UI is now located by default at <JobHistoryServer host>:19888. Jobs can be searched and viewed there just as they could in MapReduce 1.

Because the ResourceManager is meant to be agnostic to many of the concepts in MapReduce, it cannot host job information directly. Instead, it proxies to a Web UI that can. If the job is running, this proxy is the relevant MapReduce Application Master; if the job has completed, then this proxy is the JobHistoryServer. Thus, the user experience is similar to that of MapReduce 1, but the information is now coming from different places.

Summary of Configuration Changes

The following tables summarize the changes in configuration parameters between MRv1 and MRv2.

JobTracker Properties and ResourceManager Equivalents

MRv1	YARN / MRv2
`mapred.jobtracker.taskScheduler`	`yarn.resourcemanager.scheduler.class`
`mapred.jobtracker.completeuserjobs.maximum`	`yarn.resourcemanager.max-completed-applications`
`mapred.jobtracker.restart.recover`	`yarn.resourcemanager.recovery.enabled`

`mapred.job.tracker`	`yarn.resourcemanager.hostname`
	or all of the following:
	`yarn.resourcemanager.address`
	`yarn.resourcemanager.scheduler.address`
	`yarn.resourcemanager.resource-tracker.address`
	`yarn.resourcemanager.admin.address`

`mapred.job.tracker.http.address`	`yarn.resourcemanager.webapp.address`
	or
	`yarn.resourcemanager.hostname`

`mapred.job.tracker.handler.count`	`yarn.resourcemanager.resource-tracker.client.thread-count`
`mapred.hosts`	`yarn.resourcemanager.nodes.include-path`
`mapred.hosts.exclude`	`yarn.resourcemanager.nodes.exclude-path`
`mapred.cluster.max.map.memory.mb`	`yarn.scheduler.maximum-allocation-mb`
`mapred.cluster.max.reduce.memory.mb`	`yarn.scheduler.maximum-allocation-mb`
`mapred.acls.enabled`	`yarn.acl.enable`
`mapreduce.cluster.acls.enabled`	`yarn.acl.enable`

JobTracker Properties and JobHistoryServer Equivalents

MRv1	YARN / MRv2	Comment
`mapred.job.tracker.retiredjobs.cache.size`	`mapreduce.jobhistory.joblist.cache.size`	
`mapred.job.tracker.jobhistory.lru.cache.size`	`mapreduce.jobhistory.loadedjobs.cache.size`	
`mapred.job.tracker.history.completed.location`	`mapreduce.jobhistory.done-dir`	Local FS in MR1; stored in HDFS in MR2
`hadoop.job.history.user.location`	`mapreduce.jobhistory.done-dir`	
`hadoop.job.history.location`	`mapreduce.jobhistory.done-dir`	

JobTracker Properties and MapReduce ApplicationMaster Equivalents

MRv1	YARN / MRv2	Comment
`mapreduce.jobtracker.staging.root.dir`	`yarn.app.mapreduce.am.staging-dir`	Now configurable per job

TaskTracker Properties and NodeManager Equivalents

MRv1	YARN / MRv2

`mapred.tasktracker.map.tasks.maximum`	`yarn.nodemanager.resource.memory-mb`
	and
	`yarn.nodemanager.resource.cpu-vcores`

MRv1	YARN / MRv2
`mapred.tasktracker.reduce.tasks.maximum`	`yarn.nodemanager.resource.memory-mb` and `yarn.nodemanager.resource.cpu-vcores`
`mapred.tasktracker.expiry.interval`	`yarn.nm.liveliness-monitor.expiry-interval-ms`
`mapred.tasktracker.resourcecalculatorplugin`	`yarn.nodemanager.container-monitor.resource-calculator.class`
`mapred.tasktracker.taskmemorymanager.monitoring-interval`	`yarn.nodemanager.container-monitor.interval-ms`
`mapred.tasktracker.tasks.sleeptime-before-sigkill`	`yarn.nodemanager.sleep-delay-before-sigkill.ms`
`mapred.task.tracker.task-controller`	`yarn.nodemanager.container-executor.class`
`mapred.local.dir`	`yarn.nodemanager.local-dirs`
`mapreduce.cluster.local.dir`	`yarn.nodemanager.local-dirs`
`mapred.disk.healthChecker.interval`	`yarn.nodemanager.disk-health-checker.interval-ms`
`mapred.healthChecker.script.path`	`yarn.nodemanager.health-checker.script.path`
`mapred.healthChecker.interval`	`yarn.nodemanager.health-checker.interval-ms`
`mapred.healthChecker.script.timeout`	`yarn.nodemanager.health-checker.script.timeout-ms`
`mapred.healthChecker.script.args`	`yarn.nodemanager.health-checker.script.opts`
`local.cache.size`	`yarn.nodemanager.localizer.cache.target-size-mb`
`mapreduce.tasktracker.cache.local.size`	`yarn.nodemanager.localizer.cache.target-size-mb`

TaskTracker Properties and Shuffle Service Equivalents

The table that follows shows TaskTracker properties and their equivalents in the auxiliary shuffle service that runs inside NodeManagers.

MRv1	YARN / MRv2
`tasktracker.http.threads`	`mapreduce.shuffle.max.threads`
`mapred.task.tracker.http.address`	`mapreduce.shuffle.port`
`mapred.tasktracker.indexcache.mb`	`mapred.tasktracker.indexcache.mb`

Per-Job Configuration Properties

Many of these properties have new names in MRv2, but the MRv1 names will work for all properties except `mapred.job.restart.recover`.

MRv1	YARN / MRv2	Comment
`io.sort.mb`	`mapreduce.task.io.sort.mb`	MRv1 name still works
`io.sort.factor`	`mapreduce.task.io.sort.factor`	MRv1 name still works
`io.sort.spill.percent`	`mapreduce.task.io.sort.spill.percent`	MRv1 name still works
`mapred.map.tasks`	`mapreduce.job.maps`	MRv1 name still works

MRv1	YARN / MRv2	Comment
mapred.reduce.tasks	mapreduce.job.reduces	MRv1 name still works
mapred.job.map.memory.mb	mapreduce.map.memory.mb	MRv1 name still works
mapred.job.reduce.memory.mb	mapreduce.reduce.memory.mb	MRv1 name still works
mapred.map.child.log.level	mapreduce.map.log.level	MRv1 name still works
mapred.reduce.child.log.level	mapreduce.reduce.log.level	MRv1 name still works
mapred.inmem.merge.threshold	mapreduce.reduce.shuffle.merge.inmem.threshold	MRv1 name still works
mapred.job.shuffle.merge.percent	mapreduce.reduce.shuffle.merge.percent	MRv1 name still works
mapred.job.shuffle.input.buffer.percent	mapreduce.reduce.shuffle.input.buffer.percent	MRv1 name still works
mapred.job.reduce.input.buffer.percent	mapreduce.reduce.input.buffer.percent	MRv1 name still works
mapred.map.tasks.speculative.execution	mapreduce.map.speculative	Old one still works
mapred.reduce.tasks.speculative.execution	mapreduce.reduce.speculative	MRv1 name still works
mapred.min.split.size	mapreduce.input.fileinputformat.split.minsize	MRv1 name still works
keep.failed.task.files	mapreduce.task.files.preserve.failedtasks	MRv1 name still works
mapred.output.compress	mapreduce.output.fileoutputformat.compress	MRv1 name still works
mapred.map.output.compression.codec	mapreduce.map.output.compress.codec	MRv1 name still works
mapred.compress.map.output	mapreduce.map.output.compress	MRv1 name still works
mapred.output.compression.type	mapreduce.output.fileoutputformat.compress.type	MRv1 name still works
mapred.userlog.limit.kb	mapreduce.task.userlog.limit.kb	MRv1 name still works
jobclient.output.filter	mapreduce.client.output.filter	MRv1 name still works
jobclient.completion.poll.interval	mapreduce.client.completion.pollinterval	MRv1 name still works
jobclient.progress.monitor.poll.interval	mapreduce.client.progressmonitor.pollinterval	MRv1 name still works
mapred.task.profile	mapreduce.task.profile	MRv1 name still works

MRv1	YARN / MRv2	Comment
mapred.task.profile.maps	mapreduce.task.profile.maps	MRv1 name still works
mapred.task.profile.reduces	mapreduce.task.profile.reduces	MRv1 name still works
mapred.line.input.format.linespermap	mapreduce.input.lineinputformat.linespermap	MRv1 name still works
mapred.skip.attempts.to.start.skipping	mapreduce.task.skip.start.attempts	MRv1 name still works
mapred.skip.map.auto.incr.proc.count	mapreduce.map.skip.proc.count.autoincr	MRv1 name still works
mapred.skip.reduce.auto.incr.proc.count	mapreduce.reduce.skip.proc.count.autoincr	MRv1 name still works
mapred.skip.out.dir	mapreduce.job.skip.outdir	MRv1 name still works
mapred.skip.map.max.skip.records	mapreduce.map.skip.maxrecords	MRv1 name still works
mapred.skip.reduce.max.skip.groups	mapreduce.reduce.skip.maxgroups	MRv1 name still works
job.end.retry.attempts	mapreduce.job.end-notification.retry.attempts	MRv1 name still works
job.end.retry.interval	mapreduce.job.end-notification.retry.interval	MRv1 name still works
job.end.notification.url	mapreduce.job.end-notification.url	MRv1 name still works
mapred.merge.recordsBeforeProgress	mapreduce.task.merge.progress.records	MRv1 name still works
mapred.job.queue.name	mapreduce.job.queuename	MRv1 name still works
mapred.reduce.slowstart.completed.maps	mapreduce.job.reduce.slowstart.completedmaps	MRv1 name still works
mapred.map.max.attempts	mapreduce.map.maxattempts	MRv1 name still works
mapred.reduce.max.attempts	mapreduce.reduce.maxattempts	MRv1 name still works
mapred.reduce.parallel.copies	mapreduce.reduce.shuffle.parallelcopies	MRv1 name still works
mapred.task.timeout	mapreduce.task.timeout	MRv1 name still works
mapred.max.tracker.failures	mapreduce.job.maxtaskfailures.per.tracker	MRv1 name still works
mapred.job.restart.recover	mapreduce.am.max-attempts	

MRv1	YARN / MRv2	Comment
`mapred.combine.recordsBeforeProgress`	`mapreduce.task.combine.progress.records`	MRv1 name should still work - see MAPREDUCE-5130

Miscellaneous Properties

MRv1	YARN / MRv2
`mapred.heartbeats.in.second`	`yarn.resourcemanager.nodemanagers.heartbeat-interval-ms`
`mapred.userlog.retain.hours`	`yarn.log-aggregation.retain-seconds`

MRv1 Properties that have no MRv2 Equivalents

MRv1	Comment
`mapreduce.tasktracker.group`	
`mapred.child.ulimit`	
`mapred.tasktracker.dns.interface`	
`mapred.tasktracker.dns.nameserver`	
`mapred.tasktracker.instrumentation`	NodeManager does not accept instrumentation
`mapred.job.reuse.jvm.num.tasks`	JVM reuse no longer supported
`mapreduce.job.jvm.numtasks`	JVM reuse no longer supported
`mapred.task.tracker.report.address`	No need for this, as containers do not use IPC with NodeManagers, and AM ports are chosen at runtime
`mapreduce.task.tmp.dir`	No longer configurable. Now always `tmp/` (under container's local dir)
`mapred.child.tmp`	No longer configurable. Now always `tmp/` (under container's local dir)
`mapred.temp.dir`	
`mapred.jobtracker.instrumentation`	ResourceManager does not accept instrumentation
`mapred.jobtracker.plugins`	ResourceManager does not accept plugins
`mapred.task.cache.level`	
`mapred.queue.names`	These go in the scheduler-specific configuration files
`mapred.system.dir`	
`mapreduce.tasktracker.cache.local.numberdirectories`	
`mapreduce.reduce.input.limit`	
`io.sort.record.percent`	Tuned automatically (MAPREDUCE-64)
`mapred.cluster.map.memory.mb`	Not necessary; MRv2 uses resources instead of slots
`mapred.cluster.reduce.memory.mb`	Not necessary; MRv2 uses resources instead of slots
`mapred.max.tracker.blacklists`	
`mapred.jobtracker.maxtasks.per.job`	Related configurations go in scheduler-specific configuration files

MRv1	Comment
`mapred.jobtracker.taskScheduler.maxRunningTasksPerJob`	Related configurations go in scheduler-specific configuration files
`io.map.index.skip`	
`mapred.user.jobconf.limit`	
`mapred.local.dir.minspacestart`	
`mapred.local.dir.minspacekill`	
`hadoop.rpc.socket.factory.class.JobSubmissionProtocol`	
`mapreduce.tasktracker.outofband.heartbeat`	Always on
`mapred.jobtracker.job.history.block.size`	

Upgrading from CDH 4 to CDH 5

Note:

If you are using Cloudera Manager to manage CDH, do not use the instructions in this section.

- If you are running Cloudera Manager 4.x, **you must upgrade Cloudera Manager to version 5 first**, as Cloudera Manager 4 cannot manage CDH 5; see Upgrading Cloudera Manager.
- Follow directions in the current version of Managing Clusters with Cloudera Managerto upgrade CDH 4 to CDH 5 in a Cloudera-managed deployment.

Use the following information and instructions to upgrade to the latest CDH 5 release from a CDH 4 release.:

- Before You Begin
- Upgrading to CDH 5

Important:

This involves uninstalling the CDH 4 packages and installing the CDH 5 packages.

Note:

If you are migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN), see Migrating from MapReduce v1 (MRv1) to MapReduce v2 (MRv2, YARN) on page 43 for important information and instructions.

Before You Begin

Important:

Before upgrading, be sure to read about the latest Incompatible Changes and Known Issues in CDH 5 in the CDH 5 Release Notes. If you are currently running MRv1, you should read CDH 5 and MapReduce on page 25 before proceeding.

Plan Downtime

If you are upgrading a cluster that is part of a production system, be sure to plan ahead. As with any operational work, be sure to reserve a maintenance window with enough extra time allotted in case of complications. The Hadoop upgrade process is well understood, but it is best to be cautious. For production clusters, Cloudera recommends allocating up to a full day maintenance window to perform the upgrade, depending on the number of hosts, the amount of experience you have with Hadoop and Linux, and the particular hardware you are using.

Delete Symbolic Links in HDFS

If there are symbolic links in HDFS when you upgrade from CDH 4 to CDH 5, the upgrade will fail and you will have to downgrade to CDH 4, delete the symbolic links, and start over. To prevent this, proceed as follows.

To check for symbolic links in CDH 4 HDFS:

1. `cd` to the directory on the NameNode that contains the latest `fsimage` The location of this directory is specified as the value of `dfs.namenode.name.dir` (or `dfs.name.dir`) in `hdfs-site.xml`.

2. Use a command such as the following to write out the path names in the `fsimage`:

```
$ hdfs oiv -i FSIMAGE -o /tmp/YYYY-MM-DD_FSIMAGE.txt
```

3. Use a command such as the following to find the path names of any symbolic links listed in `/tmp/YYYY-MM-DD_FSIMAGE.txt` and write them out to the file `/tmp/symlinks.txt`:

```
$ grep -- "->" /tmp/YYYY-MM-DD_FSIMAGE.txt > /tmp/symlinks.txt
```

4. Delete any symbolic links listed in `/tmp/symlinks.txt`.

Considerations for Secure Clusters

If you are upgrading a cluster that has Kerberos security enabled, you must do the following:

- Before starting the upgrade, read the CDH 5 Security Guide.
- Before shutting down Hadoop services, put the NameNode into safe mode and perform a `saveNamespace` operation; see the instructions on backing up the metadata.

High Availability

In CDH 5 you can configure high availability both for the NameNode and the JobTracker or Resource Manager.

- For more information and instructions on setting up a new HA configuration, see the CDH 5 High Availability Guide.

  Important:

  If you decide to configure HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`. After completing the HDFS HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

- To upgrade an existing configuration, follow the instructions under Upgrading to CDH 5 on page 58.

Upgrading to CDH 5

Note: Are you on the right page?

Use the instructions on this page only to upgrade from CDH 4.

To upgrade from an earlier CDH 5 release to the latest version:

- Use these instructions to upgrade from a CDH 5 Beta release;
- Use these instructions to upgrade from CDH 5.0.0 or later.

Important:

1. To upgrade from CDH 4, you must uninstall CDH 4, and then install CDH 5. Make sure you allow sufficient time for this, and do the necessary backup and preparation as described below.
2. If you have configured HDFS HA with NFS shared storage, do not proceed. This configuration is not supported on CDH 5; Quorum-based storage is the only supported HDFS HA configuration on CDH 5. Unconfigure your NFS shared storage configuration before you attempt to upgrade.

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Back Up Configuration Data and Stop Services

1. Put the NameNode into safe mode and save the `fsimage`:
   a. Put the NameNode (or active NameNode in an HA configuration) into safe mode:
   ```
   $ sudo -u hdfs hdfs dfsadmin -safemode enter
   ```
   b. Perform a `saveNamespace` operation:
   ```
   $ sudo -u hdfs hdfs dfsadmin -saveNamespace
   ```
   This will result in a new `fsimage` being written out with no edit log entries.
   c. With the NameNode still in safe mode, shut down all services as instructed below.
2. For each component you are using, back up configuration data, databases, and other important files.
3. Shut down the Hadoop services across your entire cluster:
   ```
   for x in `cd /etc/init.d ; ls hadoop-*` ; do sudo service $x stop ; done
   ```
4. Check each host to make sure that there are no processes running as the `hdfs` or `mapred` users from root:
   ```
   # ps -aef | grep java
   ```

Back up the HDFS Metadata

Important:

Do this step when you are sure that all Hadoop services have been shut down. **It is particularly important that the NameNode service is not running so that you can make a consistent backup.**

To back up the HDFS metadata on the NameNode machine:

Note:

- Cloudera recommends backing up HDFS metadata on a regular basis, as well as before a major upgrade.
- `dfs.name.dir` is deprecated but still works; `dfs.namenode.name.dir` is preferred. This example uses `dfs.name.dir`.

1. Find the location of your `dfs.name.dir` (or `dfs.namenode.name.dir`); for example:
   ```
   $ grep -C1 dfs.name.dir /etc/hadoop/conf/hdfs-site.xml
   ```

You should see something like this:

```
<property>
<name>dfs.name.dir</name>
<value>/mnt/hadoop/hdfs/name</value>
```

2. Back up the directory. The path inside the <value> XML element is the path to your HDFS metadata. If you see a comma-separated list of paths, there is no need to back up all of them; they store the same data. Back up the first directory, for example, by using the following commands:

```
$ cd /mnt/hadoop/hdfs/name
# tar -cvf /root/nn_backup_data.tar .
./
./current/
./current/fsimage
./current/fstime
./current/VERSION
./current/edits
./image/
./image/fsimage
```

> **Warning:**
>
> If you see a file containing the word `lock`, the NameNode is probably still running. Repeat the preceding steps, starting by shutting down the Hadoop services.

Update Alternatives

On each node in the cluster:

1. Update the `alternatives`, for example:

```
$ sudo update-alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50
```

2. Verify that the operation succeeded:

```
$ sudo alternatives --display hadoop-conf
```

Uninstall the CDH 4 Version of Hadoop

> **Warning:**
>
> Do not proceed before you have backed up the HDFS metadata, and the files and databases for the individual components, as instructed in the previous steps.

To uninstall Hadoop:

Run this command on each host:

On Red Hat-compatible systems:

```
$ sudo yum remove  bigtop-utils bigtop-jsvc bigtop-tomcat sqoop2-client hue-common solr
```

On SLES systems:

```
$ sudo zypper remove bigtop-utils bigtop-jsvc bigtop-tomcat sqoop2-client hue-common
solr
```

On Ubuntu systems:

```
sudo apt-get remove bigtop-utils bigtop-jsvc bigtop-tomcat sqoop2-client hue-common
solr
```

Remove CDH 4 Repository Files

Remove all Cloudera CDH 4 repository files. For example, on a Red Hat or similar system, remove all files in `/etc/yum.repos.d` that have `cloudera` as part of the name.

Important:

- Before removing the files, make sure you have not added any custom entries that you want to preserve. (To preserve custom entries, back up the files before removing them.)
- Make sure you remove Impala and Search repository files, as well as the CDH repository file.

Download the Latest Version of CDH 5

Note:

For instructions on how to add a CDH 5 yum repository or build your own CDH 5 yum repository, see Installing CDH 5 on page 27.

On Red Hat-compatible systems:

1. Download the CDH 5 "1-click Install" package.

   Click the entry in the table below that matches your Red Hat or CentOS system, choose **Save File**, and save the file to a directory to which you have write access (it can be your home directory).

OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS/Oracle 6	Red Hat/CentOS/Oracle 6 link

2. Install the RPM:

   - **Red Hat/CentOS/Oracle 5**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

   - **Red Hat/CentOS/Oracle 6**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo yum clean all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

Now (optionally) add a repository key:

- **For Red Hat/CentOS/Oracle 5 systems:**

  ```
  $ sudo rpm --import
  http://archive.cloudera.com/cdh5/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
  ```

- **For Red Hat/CentOS/Oracle 6 systems:**

  ```
  $ sudo rpm --import
  http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
  ```

On SLES systems:

1. Download the CDH 5 "1-click Install" package.

   Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).

2. Install the RPM:

   ```
   $ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
   ```

3. Update your system package index by running:

   ```
   $ sudo zypper refresh
   ```

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo zypper clean --all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

Now (optionally) add a repository key:

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

On Ubuntu and Debian systems:

1. Download the CDH 5 "1-click Install" package:

OS Version	Click this Link
Wheezy	Wheezy link
Precise	Precise link

2. Install the package. Do one of the following:

   - Choose **Open with** in the download window to use the package manager.
   - Choose **Save File**, save the package to a directory to which you have write access (it can be your home directory) and install it from the command line, for example:

     ```
     sudo dpkg -i cdh5-repository_1.0_all.deb
     ```

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo apt-get update
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

Now (optionally) add a repository key:

- **For Ubuntu Precise systems:**

```
$ curl -s http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Debian Wheezy systems:**

```
$ curl -s http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh/archive.key
| sudo apt-key add -
```

Install CDH 5 with YARN

Note:

Skip this step and go to Install CDH 5 with MRv1 on page 65 if you intend to use *only* MRv1.

1. Install and deploy ZooKeeper.

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
Resource Manager host (analogous to MRv1 JobTracker) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager`
NameNode host(s) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`

Where to install	Install commands
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the Resource Manager running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
SLES	`sudo zypper clean --all; sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
One host in the cluster running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
All client hosts, running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-client`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-client`

Note:

The `hadoop-yarn` and `hadoop-hdfs` packages are installed on each system automatically as dependencies of the other packages.

3. If you are installing Llama, make sure that `hadoop.proxyuser.llama.hosts` and `hadoop.proxyuser.llama.groups` are configured in your `core-site.xml` as follows:

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

Install CDH 5 with MRv1

Note:

Skip this step if you intend to use *only* YARN. If you are installing both YARN and MRv1, you can skip any packages you have already installed in Step 6a.

To install CDH 5 with MRv1:

Note:

If you are also installing YARN, you can skip any packages you have already installed in Step 6a.

1. Install and deploy ZooKeeper.

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
JobTracker host running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-jobtracker`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-jobtracker`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-jobtracker`
NameNode host(s) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`

Where to install	Install commands
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the JobTracker, NameNode, and Secondary (or Standby) NameNode hosts, running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
All client hosts, running:	
Red Hat/CentOS compatible	`sudo yum clean all; sudo yum install hadoop-client`
SLES	`sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-client`

Copy the CDH 5 Logging File

Copy over the `log4j.properties` file to your custom directory on each node in the cluster; for example:

```
$ cp /etc/hadoop/conf.empty/log4j.properties /etc/hadoop/conf.my_cluster/log4j.properties
```

In an HA Deployment, Upgrade and Start the Journal Nodes

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 proceeding to the next step.

Important:

The JournalNodes must be up and running CDH 5 before you proceed.

Upgrade the HDFS Metadata

Note:

What you do in this step differs depending on whether you are upgrading an HDFS HA deployment using Quorum-based storage, or a non-HA deployment using a secondary NameNode. (If you have an HDFS HA deployment using NFS storage, do not proceed; you cannot upgrade that configuration to CDH 5. Unconfigure your NFS shared storage configuration before you attempt to upgrade.)

- For an HA deployment, do sub-steps 1, 2, and 3.
- For a non-HA deployment, do sub-steps 1, 3, and 4.

1. To upgrade the HDFS metadata, run the following command on the NameNode. If HA is enabled, do this on the *active NameNode only*, and make sure the JournalNodes have been upgraded to CDH 5 and are up and running before you run the command.

```
$ sudo service hadoop-hdfs-namenode -upgrade
```

Important:

In an HDFS HA deployment, it is critically important that you do this on only one NameNode.

You can watch the progress of the upgrade by running:

```
$ sudo tail -f /var/log/hadoop-hdfs/hadoop-hdfs-namenode-<hostname>.log
```

Look for a line that confirms the upgrade is complete, such as:
`/var/lib/hadoop-hdfs/cache/hadoop/dfs/<name> is complete`

Note:

The NameNode upgrade process can take a while depending on how many files you have.

2. *Do this step only in an HA configuration. Otherwise skip to starting up the DataNodes.*

Wait for NameNode to exit safe mode, and then re-start the standby NameNode.

- If Kerberos is enabled:

```
$ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM>
 && hdfs namenode -bootstrapStandby
```

```
$ sudo service hadoop-hdfs-namenode start
```

- If Kerberos is not enabled:

```
$ sudo -u hdfs hdfs namenode -bootstrapStandby
$ sudo service hadoop-hdfs-namenode start
```

For more information about the `haadmin -failover` command, see Administering an HDFS High Availability Cluster.

3. Start up the DataNodes:

   On each DataNode:

```
$ sudo service hadoop-hdfs-datanode start
```

4. *Do this step only in a non-HA configuration. Otherwise skip to starting YARN or MRv1.*

   Wait for NameNode to exit safe mode, and then start the Secondary NameNode.

   a. To check that the NameNode has exited safe mode, look for messages in the log file, or the NameNode's web interface, that say "`...no longer in safe mode.`"
   b. To start the Secondary NameNode (if used), enter the following command on the Secondary NameNode host:

```
$ sudo service hadoop-hdfs-secondarynamenode start
```

   c. To complete the cluster upgrade, follow the remaining steps below.

Start YARN or MapReduce MRv1

You are now ready to start and test MRv1 or YARN.

For YARN	or For MRv1
Start YARN and the MapReduce JobHistory Server	Start MRv1
Verify basic cluster operation	Verify basic cluster operation

Start MapReduce with YARN

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 10a and 10b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start YARN. First, create directories and set the correct permissions.

For more information see Deploying MapReduce v2 (YARN) on a Cluster.

Create a history directory and set permissions; for example:

```
sudo -u hdfs hadoop fs -mkdir /user/history
sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
sudo -u hdfs hadoop fs -chown yarn /user/history
```

Create the `/var/log/hadoop-yarn` directory and set ownership:

```
$ sudo -u hdfs hadoop fs -mkdir /var/log/hadoop-yarn
$ sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn
```

You need to create this directory because it is the parent of `/var/log/hadoop-yarn/apps` which is explicitly configured in the `yarn-site.xml`.

Verify the directory structure, ownership, and permissions:

```
$ sudo -u hdfs hadoop fs -ls -R /
```

You should see:

```
drwxrwxrwt   - hdfs supergroup          0 2012-04-19 14:31 /tmp
drwxr-xr-x   - hdfs supergroup          0 2012-05-31 10:26 /user
drwxrwxrwt   - yarn supergroup          0 2012-04-19 14:31 /user/history
drwxr-xr-x   - hdfs   supergroup        0 2012-05-31 15:31 /var
drwxr-xr-x   - hdfs   supergroup        0 2012-05-31 15:31 /var/log
drwxr-xr-x   - yarn   mapred            0 2012-05-31 15:31 /var/log/hadoop-yarn
```

To start YARN, start the ResourceManager and NodeManager services:

Note:

Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

```
$ sudo service hadoop-yarn-resourcemanager start
```

On each NodeManager system (typically the same ones where DataNode service runs):

```
$ sudo service hadoop-yarn-nodemanager start
```

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

```
$ sudo service hadoop-mapreduce-historyserver start
```

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

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

Verify basic cluster operation for YARN.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

> **Note:**
>
> For important configuration information, see Deploying MapReduce v2 (YARN) on a Cluster.

1. Create a home directory on HDFS for the user who will be running the job (for example, joe):

   ```
   $ sudo -u hdfs hadoop fs -mkdir /user/joe
   $ sudo -u hdfs hadoop fs -chown joe /user/joe
   ```

   Do the following steps as the user joe.

2. Make a directory in HDFS called input and copy some XML files into it by running the following commands in pseudo-distributed mode:

   ```
   $ hadoop fs -mkdir input
   $ hadoop fs -put /etc/hadoop/conf/*.xml input
   $ hadoop fs -ls input
   Found 3 items:
   -rw-r--r--   1 joe supergroup       1348 2012-02-13 12:21 input/core-site.xml
   -rw-r--r--   1 joe supergroup       1913 2012-02-13 12:21 input/hdfs-site.xml
   -rw-r--r--   1 joe supergroup       1001 2012-02-13 12:21 input/mapred-site.xml
   ```

3. Set HADOOP_MAPRED_HOME for user joe:

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

4. Run an example Hadoop job to grep with a regular expression in your input data.

   ```
   $ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input
   output23 'dfs[a-z.]+'
   ```

5. After the job completes, you can find the output in the HDFS directory named output23 because you specified that output directory to Hadoop.

   ```
   $ hadoop fs -ls
   Found 2 items
   drwxr-xr-x   - joe supergroup  0 2009-08-18 18:36 /user/joe/input
   drwxr-xr-x   - joe supergroup  0 2009-08-18 18:38 /user/joe/output23
   ```

   You can see that there is a new directory called output23.

6. List the output files.

   ```
   $ hadoop fs -ls output23
   Found 2 items
   drwxr-xr-x  -  joe supergroup     0 2009-02-25 10:33   /user/joe/output23/_SUCCESS
   -rw-r--r--  1  joe supergroup  1068 2009-02-25 10:33
   /user/joe/output23/part-r-00000
   ```

7. Read the results in the output file.

   ```
   $ hadoop fs -cat output23/part-r-00000 | head
   1    dfs.safemode.min.datanodes
   1    dfs.safemode.extension
   1    dfs.replication
   1    dfs.permissions.enabled
   1    dfs.namenode.name.dir
   1    dfs.namenode.checkpoint.dir
   1    dfs.datanode.data.dir
   ```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Start MapReduce (MRv1)

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 9a and 9b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start MapReduce. On each TaskTracker system:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

On the JobTracker system:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

Verify that the JobTracker and TaskTracker started properly.

```
$ sudo jps | grep Tracker
```

If the permissions of directories are not configured correctly, the JobTracker and TaskTracker processes start and immediately fail. If this happens, check the JobTracker and TaskTracker logs and set the permissions correctly.

Important:

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

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

Verify basic cluster operation for MRv1.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

1. Create a home directory on HDFS for the user who will be running the job (for example, `joe`):

   ```
   $ sudo -u hdfs hadoop fs -mkdir /user/joe
   $ sudo -u hdfs hadoop fs -chown joe /user/joe
   ```

   Do the following steps as the user `joe`.

2. Make a directory in HDFS called `input` and copy some XML files into it by running the following commands:

   ```
   $ hadoop fs -mkdir input
   $ hadoop fs -put /etc/hadoop/conf/*.xml input
   $ hadoop fs -ls input
   Found 3 items:
   -rw-r--r--   1 joe supergroup       1348 2012-02-13 12:21 input/core-site.xml
   -rw-r--r--   1 joe supergroup       1913 2012-02-13 12:21 input/hdfs-site.xml
   -rw-r--r--   1 joe supergroup       1001 2012-02-13 12:21 input/mapred-site.xml
   ```

3. Set `HADOOP_MAPRED_HOME` for user `joe`:

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

4. Run an example Hadoop job to grep with a regular expression in your input data.

```
$ /usr/bin/hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep input
 output 'dfs[a-z.]+'
```

5. After the job completes, you can find the output in the HDFS directory named `output` because you specified that output directory to Hadoop.

```
$ hadoop fs -ls
Found 2 items
drwxr-xr-x   - joe supergroup  0 2009-08-18 18:36 /user/joe/input
drwxr-xr-x   - joe supergroup  0 2009-08-18 18:38 /user/joe/output
```

You can see that there is a new directory called `output`.

6. List the output files.

```
$ hadoop fs -ls output
Found 2 items
drwxr-xr-x  -  joe supergroup     0 2009-02-25 10:33   /user/joe/output/_logs
-rw-r--r--  1  joe supergroup  1068 2009-02-25 10:33   /user/joe/output/part-00000
-rw-r--r-   1  joe supergroup     0 2009-02-25 10:33   /user/joe/output/_SUCCESS
```

7. Read the results in the output file; for example:

```
$ hadoop fs -cat output/part-00000 | head
1       dfs.datanode.data.dir
1       dfs.namenode.checkpoint.dir
1       dfs.namenode.name.dir
1       dfs.replication
1       dfs.safemode.extension
1       dfs.safemode.min.datanodes
```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Set the Sticky Bit

For security reasons Cloudera strongly recommends you set the sticky bit on directories if you have not already done so.

The sticky bit prevents anyone except the superuser, directory owner, or file owner from deleting or moving the files within a directory. (Setting the sticky bit for a file has no effect.) Do this for directories such as `/tmp`. (For instructions on creating `/tmp` and setting its permissions, see these instructions).

Re-Install CDH 5 Components

CDH 5 Components

Use the following sections to install or upgrade CDH 5 components:

- Crunch Installation on page 155
- Flume Installation on page 157

- HBase Installation on page 169
- Installing and Using HCatalog on page 203
- Hive Installation on page 233
- HttpFS Installation on page 259
- Hue Installation on page 263
- Impala Installation on page 211
- Llama Installation on page 291
- Mahout Installation on page 293
- Oozie Installation on page 297
- Pig Installation on page 319
- Search Installation on page 325
- Sentry Installation on page 327
- Snappy Installation on page 329
- Spark Installation on page 333
- Sqoop 1 Installation on page 339
- Sqoop 2 Installation on page 345
- Whirr Installation on page 353
- ZooKeeper Installation

See also the instructions for installing or updating LZO.

Apply Configuration File Changes

Important:

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH 4 configuration file to the new CDH 5 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see Automatic handling of configuration files by `dpkg`.

For example, if you have modified your CDH 4 `zoo.cfg` configuration file (`/etc/zookeeper.dist/zoo.cfg`), RPM uninstall and re-install (using `yum remove`) renames and preserves a copy of your modified `zoo.cfg` as `/etc/zookeeper.dist/zoo.cfg.rpmsave`. You should compare this to the new `/etc/zookeeper/conf/zoo.cfg` and resolve any differences that should be carried forward (typically where you have changed property value defaults). Do this for each component you upgrade to CDH 5.

Finalize the HDFS Metadata Upgrade

To finalize the HDFS metadata upgrade you began earlier in this procedure, proceed as follows:

1. Make sure you are satisfied that the CDH 5 upgrade has succeeded and everything is running smoothly. This could take a matter of days, or even weeks.

   Warning:

   Do not proceed until you are sure you are satisfied with the new deployment. Once you have finalized the HDFS metadata, you cannot revert to an earlier version of HDFS.

   Note:

   If you need to restart the NameNode during this period (after having begun the upgrade process, but before you've run `finalizeUpgrade`) simply restart your NameNode without the `-upgrade` option.

2. Finalize the HDFS metadata upgrade: use one of the following commands, depending on whether Kerberos is enabled (see Configuring Hadoop Security in CDH 5).

   > **Important:**
   >
   > In an HDFS HA deployment, make sure that both the NameNodes and all of the JournalNodes are up and functioning normally before you proceed.

   - If Kerberos is enabled:

     ```
     $ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM>
      && hdfs dfsadmin -finalizeUpgrade
     ```

   - If Kerberos is not enabled:

     ```
     $ sudo -u hdfs hdfs dfsadmin -finalizeUpgrade
     ```

   > **Note:**
   >
   > After the metadata upgrade completes, the `previous/` and `blocksBeingWritten/` directories in the DataNodes' data directories aren't cleared until the DataNodes are restarted.

Upgrading from an Earlier CDH 5 Release to the Latest Release

Important:

If you are using Cloudera Manager to manage CDH, do not use the instructions in this section. Follow the directions in Managing Clusters with Cloudera Manager to upgrade to the latest version of CDH 5 in a Cloudera-managed deployment.

Important:

- **Use the right instructions:** the instructions in this section describe how to upgrade to the latest CDH 5 release from an earlier CDH 5 release. If you are upgrading from a CDH 4 release, use the instructions under Upgrading from CDH 4 to CDH 5 on page 57 instead.
- **MapReduce v1 (MRv1) and MapReduce v2 (YARN):** this page covers upgrade for MapReduce v1 (MRv1) and MapReduce v2 (YARN). MRv1 and YARN share common configuration files, so it is safe to *configure* both of them so long as you *run* only one set of daemons at any one time. Cloudera **does not** support running MRv1 and YARN daemons on the same nodes at the same time; it will degrade performance and may result in an unstable cluster deployment. Before deciding to deploy YARN, make sure you read the discussion on the CDH 5 Installation page under MapReduce 2.0 (YARN).

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

The following sections provide the information and instructions you need:

- Before You Begin
- Upgrading from a CDH 5 Beta Release to the Latest Version on page 76
- Upgrading from CDH 5.0.0 or Later to the Latest Release on page 91

Before You Begin

Note:

- Before upgrading, be sure to read about the latest Incompatible Changes and Known Issues and Workarounds in CDH 5 in the CDH 5 Release Notes.
- If you are upgrading a cluster that is part of a production system, be sure to plan ahead. As with any operational work, be sure to reserve a maintenance window with enough extra time allotted in case of complications. The Hadoop upgrade process is well understood, but it is best to be cautious. For production clusters, Cloudera recommends allocating up to a full day maintenance window to perform the upgrade, depending on the number of hosts, the amount of experience you have with Hadoop and Linux, and the particular hardware you are using.

Upgrading from an Earlier CDH 5 Release to the Latest Release

- The instructions in this section assume you are upgrading a multi-node cluster. If you are running a pseudo-distributed (single-machine) cluster, Cloudera recommends that you copy your data off the cluster, remove the old CDH release, install Hadoop from CDH 5, and then restore your data.
- If you have a multi-node cluster running an earlier version of CDH 5, use the instructions in one of the following sections to upgrade your cluster to the latest version:
  - Upgrading from a CDH 5 Beta Release to the Latest Version on page 76
  - Upgrading from CDH 5.0.0 or Later to the Latest Release on page 91

Upgrading from a CDH 5 Beta Release to the Latest Version

Use the instructions that follow to upgrade to the latest version of CDH 5.

Note: Are you on the right page?

Use the instructions on this page only to upgrade from a CDH 5 Beta release.

If you are not currently running a CDH 5 Beta release:

- Use these instructions to upgrade from CDH 5.0.0 or later;
- Use these instructions to upgrade from a CDH 4 release.

Step 1: Prepare the cluster for the upgrade

1. Put the NameNode into safe mode and save the`fsimage`:

   a. Put the NameNode (or active NameNode in an HA configuration) into safe mode:

   ```
   $ sudo -u hdfs hdfs dfsadmin -safemode enter
   ```

   b. Perform a `saveNamespace` operation:

   ```
   $ sudo -u hdfs hdfs dfsadmin -saveNamespace
   ```

   This will result in a new `fsimage` being written out with no edit log entries.

   c. With the NameNode still in safe mode, shut down all services as instructed below.

2. Shut down Hadoop services across your entire cluster by running the following command on every host in your cluster:

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

3. 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
   ```

   Important:

   When you are sure that all Hadoop services have been shut down, do the following step. **It is particularly important that the NameNode service is not running so that you can make a consistent backup**.

4. Back up the HDFS metadata on the NameNode machine, as follows.

Note:

- Cloudera recommends backing up HDFS metadata on a regular basis, as well as before a major upgrade.
- `dfs.name.dir` is deprecated but still works; `dfs.namenode.name.dir` is preferred. This example uses `dfs.name.dir`.

a. Find the location of your `dfs.name.dir` (or `dfs.namenode.name.dir`); for example:

```
$ grep -C1 dfs.name.dir /etc/hadoop/conf/hdfs-site.xml
<property> <name>dfs.name.dir</name> <value>/mnt/hadoop/hdfs/name</value>
</property>
```

b. Back up the directory. The path inside the <value> XML element is the path to your HDFS metadata. If you see a comma-separated list of paths, there is no need to back up all of them; they store the same data. Back up the first directory, for example, by using the following commands:

```
$ cd /mnt/hadoop/hdfs/name
# tar -cvf /root/nn_backup_data.tar .
./
./current/
./current/fsimage
./current/fstime
./current/VERSION
./current/edits
./image/
./image/fsimage
```

Warning:

If you see a file containing the word *lock*, the NameNode is probably still running. Repeat the preceding steps from the beginning; start at Step 1 and shut down the Hadoop services.

Step 2: If necessary, download the CDH 5 "1-click"package on each of the hosts in your cluster

Before you begin: Check whether you have the CDH 5 "1-click" repository installed.

- On Red Hat/CentOS-compatible and SLES systems:

```
rpm -q CDH 5-repository
```

If you are upgrading from CDH 5 Beta 1 or later, you should see:

```
CDH5-repository-1-0
```

In this case, skip to Step 3. If instead you see:

```
package CDH 5-repository is not installed
```

proceed with this step.

- On Ubuntu and Debian systems:

```
 dpkg -l | grep CDH 5-repository
```

If the repository is installed, skip to Step 3; otherwise proceed with this step.

If the CDH 5 "1-click" repository is not already installed on each host in the cluster, follow the instructions below for that host's operating system:

Upgrading from an Earlier CDH 5 Release to the Latest Release

Instructions for Red Hat-compatible systems

Instructions for SLES systems

Instructions for Ubuntu and Debian systems

On Red Hat-compatible systems:

1. Download the CDH 5 "1-click Install" package.

   Click the entry in the table below that matches your Red Hat or CentOS system, choose **Save File**, and save the file to a directory to which you have write access (it can be your home directory).

OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS/Oracle 6	Red Hat/CentOS/Oracle 6 link

2. Install the RPM:

   - **Red Hat/CentOS/Oracle 5**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

   - **Red Hat/CentOS/Oracle 6**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

Note:

For instructions on how to add a CDH 5 yum repository or build your own CDH 5 yum repository, see Installing CDH 5 On Red Hat-compatible systems.

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo yum clean all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

On SLES systems:

1. Download the CDH 5 "1-click Install" package.

   Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).

2. Install the RPM:

```
$ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
```

3. Update your system package index by running:

   ```
   $ sudo zypper refresh
   ```

```
$ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
```

> **Note:**
>
> For instructions on how to add a repository or build your own repository, see Installing CDH 5 on SLES Systems.

> **Note: Make sure your repositories are up to date**
>
> Before proceeding, make sure the repositories on each system are up to date:
>
> ```
> sudo zypper clean --all
> ```
>
> This ensures that the system repositories contain the latest software (it does not actually install anything).

On Ubuntu and Debian systems:

1. Download the CDH 5 "1-click Install" package:

OS Version	Click this Link
Wheezy	Wheezy link
Precise	Precise link

2. Install the package. Do one of the following:
   - Choose **Open with** in the download window to use the package manager.
   - Choose **Save File**, save the package to a directory to which you have write access (it can be your home directory) and install it from the command line, for example:

     ```
     sudo dpkg -i cdh5-repository_1.0_all.deb
     ```

> **Note:**
>
> For instructions on how to add a repository or build your own repository, see Installing CDH 5 on Ubuntu Systems.

> **Note: Make sure your repositories are up to date**
>
> Before proceeding, make sure the repositories on each system are up to date:
>
> ```
> sudo apt-get update
> ```
>
> This ensures that the system repositories contain the latest software (it does not actually install anything).

Step 3: Upgrade the Packages on the Appropriate Hosts

Upgrade MRv1, YARN, or both, depending on what you intend to use.

Note:

- Remember that you can install and configure both MRv1 and YARN, but you should not run them both on the same set of nodes at the same time.
- If you are using HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`.

Before installing MRv1 or YARN: (Optionally) add a repository key on each system in the cluster, if you have not already done so. Add the Cloudera Public GPG Key to your repository by executing one of the following commands:

- **For Red Hat/CentOS/Oracle 5 systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Red Hat/CentOS/Oracle 6 systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For all SLES systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Ubuntu Precise systems:**

```
$ curl -s
http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Debian Wheezy systems:**

```
$ curl -s
http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh/archive.key
| sudo apt-key add -
```

Step 3a: If you are using MRv1, upgrade the MRv1 packages on the appropriate hosts.

Skip this step if you are using YARN exclusively. Otherwise upgrade each type of daemon package on the appropriate hosts as follows:

1. Install and deploy ZooKeeper:

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
JobTracker host running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-jobtracker`

Where to install	Install commands
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-jobtracker`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-jobtracker`
NameNode host running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the JobTracker, NameNode, and Secondary (or Standby) NameNode hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
All client hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-client`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-client`

Step 3b: If you are using YARN, upgrade the YARN packages on the appropriate hosts.

Skip this step if you are using MRv1 exclusively. Otherwise upgrade each type of daemon package on the appropriate hosts as follows:

1. Install and deploy ZooKeeper:

Important:

Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
Resource Manager host (analogous to MRv1 JobTracker) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager`
NameNode host running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the Resource Manager (analogous to MRv1 TaskTrackers) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`

Where to install	Install commands
One host in the cluster running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
All client hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-client`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-client`

Note:

The `hadoop-yarn` and `hadoop-hdfs` packages are installed on each system automatically as dependencies of the other packages.

Step 4: In an HA Deployment, Upgrade and Start the Journal Nodes

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 proceeding to the next step.

Important:

The JournalNodes must be up and running CDH 5 before you proceed.

Step 5: Upgrade the HDFS Metadata

Note:

What you do in this step differs depending on whether you are upgrading an HDFS HA deployment, or a non-HA deployment using a secondary NameNode:

- For an HA deployment, do sub-steps 1, 2, and 3.
- For a non-HA deployment, do sub-steps 1, 3, and 4.

1. To upgrade the HDFS metadata, run the following command on the NameNode. If HA is enabled, do this on the *active NameNode only*, and make sure the JournalNodes have been upgraded to CDH 5 and are up and running before you run the command.

   ```
   $ sudo service hadoop-hdfs-namenode -upgrade
   ```

   Important:

   In an HDFS HA deployment, it is critically important that you do this on only one NameNode.

   You can watch the progress of the upgrade by running:

   ```
   $ sudo tail -f /var/log/hadoop-hdfs/hadoop-hdfs-namenode-<hostname>.log
   ```

   Look for a line that confirms the upgrade is complete, such as:
   `/var/lib/hadoop-hdfs/cache/hadoop/dfs/<name> is complete`

   Note:

   The NameNode upgrade process can take a while depending on how many files you have.

2. *Do this step only in an HA configuration. Otherwise skip to starting up the DataNodes.*

   Wait for NameNode to exit safe mode, and then re-start the standby NameNode.

   - If Kerberos is enabled:

     ```
     $ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM>
      && hdfs namenode -bootstrapStandby
     ```

     ```
     $ sudo service hadoop-hdfs-namenode start
     ```

   - If Kerberos is not enabled:

     ```
     $ sudo -u hdfs hdfs namenode -bootstrapStandby
     $ sudo service hadoop-hdfs-namenode start
     ```

   For more information about the `haadmin -failover` command, see Administering an HDFS High Availability Cluster.

3. Start up the DataNodes:

On each DataNode:

```
$ sudo service hadoop-hdfs-datanode start
```

4. *Do this step only in a non-HA configuration. Otherwise skip to starting YARN or MRv1.*

   Wait for NameNode to exit safe mode, and then start the Secondary NameNode.

   a. To check that the NameNode has exited safe mode, look for messages in the log file, or the NameNode's web interface, that say `"...no longer in safe mode."`
   b. To start the Secondary NameNode (if used), enter the following command on the Secondary NameNode host:

      ```
      $ sudo service hadoop-hdfs-secondarynamenode start
      ```

   c. To complete the cluster upgrade, follow the remaining steps below.

Step 6: Start MapReduce (MRv1) or YARN

You are now ready to start and test MRv1 or YARN.

For MRv1	For YARN
Start MRv1	Start YARN and the MapReduce JobHistory Server
Verify basic cluster operation	Verify basic cluster operation

Step 6a: Start MapReduce (MRv1)

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 6a and 6b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start MapReduce. On each TaskTracker system:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

On the JobTracker system:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

Verify that the JobTracker and TaskTracker started properly.

```
$ sudo jps | grep Tracker
```

If the permissions of directories are not configured correctly, the JobTracker and TaskTracker processes start and immediately fail. If this happens, check the JobTracker and TaskTracker logs and set the permissions correctly.

Verify basic cluster operation for MRv1.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

Note:

For important configuration information, see Deploying MapReduce v1 (MRv1) on a Cluster.

1. Create a home directory on HDFS for the user who will be running the job (for example, `joe`):

   ```
   sudo -u hdfs hadoop fs -mkdir -p /user/joe sudo -u hdfs hadoop fs -chown joe
   /user/joe
   ```

   Do the following steps as the user `joe`.

2. Make a directory in HDFS called `input` and copy some XML files into it by running the following commands:

   ```
   $ hadoop fs -mkdir input
   $ hadoop fs -put /etc/hadoop/conf/*.xml input
   $ hadoop fs -ls input
   Found 3 items:
   -rw-r--r-- 1 joe supergroup 1348 2012-02-13 12:21 input/core-site.xml
   -rw-r--r-- 1 joe supergroup 1913 2012-02-13 12:21 input/hdfs-site.xml
   -rw-r--r-- 1 joe supergroup 1001 2012-02-13 12:21 input/mapred-site.xml
   ```

3. Run an example Hadoop job to grep with a regular expression in your input data.

   ```
   $ /usr/bin/hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep input
    output 'dfs[a-z.]+'
   ```

4. After the job completes, you can find the output in the HDFS directory named `output` because you specified that output directory to Hadoop.

   ```
   $ hadoop fs -ls
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:36 /user/joe/input
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:38 /user/joe/output
   ```

   You can see that there is a new directory called `output`.

5. List the output files.

   ```
   $ hadoop fs -ls output
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-02-25 10:33 /user/joe/output/_logs
   -rw-r--r-- 1 joe supergroup 1068 2009-02-25 10:33 /user/joe/output/part-00000
   -rw-r--r- 1 joe supergroup 0 2009-02-25 10:33 /user/joe/output/_SUCCESS
   ```

6. Read the results in the output file; for example:

   ```
   $ hadoop fs -cat output/part-00000 | head
   1 dfs.datanode.data.dir
   1 dfs.namenode.checkpoint.dir
   1 dfs.namenode.name.dir
   1 dfs.replication
   1 dfs.safemode.extension
   1 dfs.safemode.min.datanodes
   ```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Step 6b: Start MapReduce with YARN

> **Important:**
>
> Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 6a and 6b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start YARN. First, if you have not already done so, create directories and set the correct permissions.

> **Note:** For more information see Deploying MapReduce v2 (YARN) on a Cluster.

Create a history directory and set permissions; for example:

```
$ sudo -u hdfs hadoop fs -mkdir -p /user/history
$ sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
 $ sudo -u hdfs hadoop fs -chown yarn /user/history
```

Create the `/var/log/hadoop-yarn` directory and set ownership:

```
$ sudo -u hdfs hadoop fs -mkdir -p /var/log/hadoop-yarn
$ sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn
```

> **Note:** You need to create this directory because it is the parent of `/var/log/hadoop-yarn/apps` which is explicitly configured in the `yarn-site.xml`.

Verify the directory structure, ownership, and permissions:

```
$ sudo -u hdfs hadoop fs -ls -R /
```

You should see:

```
drwxrwxrwt - hdfs supergroup 0 2012-04-19 14:31 /tmp
drwxr-xr-x - hdfs supergroup 0 2012-05-31 10:26 /user
drwxrwxrwt - yarn supergroup 0 2012-04-19 14:31 /user/history
drwxr-xr-x - hdfs supergroup 0 2012-05-31 15:31 /var
drwxr-xr-x - hdfs supergroup 0 2012-05-31 15:31 /var/log
drwxr-xr-x - yarn mapred 0 2012-05-31 15:31 /var/log/hadoop-yarn
```

To start YARN, start the ResourceManager and NodeManager services:

> **Note:**
>
> Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

```
$ sudo service hadoop-yarn-resourcemanager start
```

On each NodeManager system (typically the same ones where DataNode service runs):

```
$ sudo service hadoop-yarn-nodemanager start
```

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

```
$ sudo service hadoop-mapreduce-historyserver start
```

For each user who will be submitting MapReduce jobs using MapReduce v2 (YARN), or running Pig, Hive, or Sqoop 1 in a YARN installation, set the `HADOOP_MAPRED_HOME` environment variable as follows:

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

Verify basic cluster operation for YARN.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

Note:

For important configuration information, see Deploying MapReduce v2 (YARN) on a Cluster.

1. Create a home directory on HDFS for the user who will be running the job (for example, `joe`):

```
$ sudo -u hdfs hadoop fs -mkdir -p /user/joe sudo -u hdfs hadoop fs -chown joe
/user/joe
```

   Do the following steps as the user `joe`.

2. Make a directory in HDFS called `input` and copy some XML files into it by running the following commands in pseudo-distributed mode:

```
$ hadoop fs -mkdir input
$ hadoop fs -put /etc/hadoop/conf/*.xml input
$ hadoop fs -ls input
Found 3 items:
-rw-r--r-- 1 joe supergroup 1348 2012-02-13 12:21 input/core-site.xml
-rw-r--r-- 1 joe supergroup 1913 2012-02-13 12:21 input/hdfs-site.xml
-rw-r--r-- 1 joe supergroup 1001 2012-02-13 12:21 input/mapred-site.xml
```

3. Set `HADOOP_MAPRED_HOME` for user `joe`:

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

4. Run an example Hadoop job to `grep` with a regular expression in your input data.

```
$ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input
output23 'dfs[a-z.]+'
```

5. After the job completes, you can find the output in the HDFS directory named `output23` because you specified that output directory to Hadoop.

```
$ hadoop fs -ls
Found 2 items
drwxr-xr-x - joe supergroup 0 2009-08-18 18:36 /user/joe/input
drwxr-xr-x - joe supergroup 0 2009-08-18 18:38 /user/joe/output23
```

   You can see that there is a new directory called `output23`.

6. List the output files:

```
$ hadoop fs -ls output23
Found 2 items
drwxr-xr-x - joe supergroup 0 2009-02-25 10:33 /user/joe/output23/_SUCCESS
-rw-r--r-- 1 joe supergroup 1068 2009-02-25 10:33 /user/joe/output23/part-r-00000
```

7. Read the results in the output file:

```
$ hadoop fs -cat output23/part-r-00000 | head
1 dfs.safemode.min.datanodes
```

```
1 dfs.safemode.extension
1 dfs.replication
1 dfs.permissions.enabled
1 dfs.namenode.name.dir
1 dfs.namenode.checkpoint.dir
1 dfs.datanode.data.dir
```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Step 7: Set the Sticky Bit

For security reasons Cloudera strongly recommends you set the sticky bit on directories if you have not already done so.

The sticky bit prevents anyone except the superuser, directory owner, or file owner from deleting or moving the files within a directory. (Setting the sticky bit for a file has no effect.) Do this for directories such as `/tmp`. (For instructions on creating `/tmp` and setting its permissions, see these instructions).

Step 8: Upgrade Components

Note:

- For important information on new and changed components, see the Release Notes. To see whether there is a new version of a particular component in CDH 5, check the CDH Version and Packaging Information.
- Cloudera recommends that you regularly update the software on each system in the cluster (for example, on a RHEL-compatible system, regularly run `yum update`) to ensure that all the dependencies for any given component are up to date. (If you have not been in the habit of doing this, be aware that the command may take a while to run the first time you use it.)

CDH 5 Components

Use the following sections to install or upgrade CDH 5 components:

- Crunch Installation on page 155
- Flume Installation on page 157
- HBase Installation on page 169
- Installing and Using HCatalog on page 203
- Hive Installation on page 233
- HttpFS Installation on page 259
- Hue Installation on page 263
- Impala Installation on page 211
- Llama Installation on page 291
- Mahout Installation on page 293
- Oozie Installation on page 297
- Pig Installation on page 319
- Search Installation on page 325
- Sentry Installation on page 327
- Snappy Installation on page 329
- Spark Installation on page 333

- Sqoop 1 Installation on page 339
- Sqoop 2 Installation on page 345
- Whirr Installation on page 353
- ZooKeeper Installation

See also the instructions for installing or updating LZO.

Step 9: Apply Configuration File Changes if Necessary

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

For example, if you have modified your `zoo.cfg` configuration file (`/etc/zookeeper/zoo.cfg`), the upgrade renames and preserves a copy of your modified `zoo.cfg` as `/etc/zookeeper/zoo.cfg.rpmsave`. If you have not already done so, you should now compare this to the new `/etc/zookeeper/conf/zoo.cfg`, resolve differences, and make any changes that should be carried forward (typically where you have changed property value defaults). Do this for each component you upgrade.

Step 10: Finalize the HDFS Metadata Upgrade

To finalize the HDFS metadata upgrade you began earlier in this procedure, proceed as follows:

1. Make sure you are satisfied that the CDH 5 upgrade has succeeded and everything is running smoothly. This could take a matter of days, or even weeks.

   Warning:

   Do not proceed until you are sure you are satisfied with the new deployment. Once you have finalized the HDFS metadata, you cannot revert to an earlier version of HDFS.

   Note:

   If you need to restart the NameNode during this period (after having begun the upgrade process, but before you've run `finalizeUpgrade`) simply restart your NameNode without the `-upgrade` option.

2. Finalize the HDFS metadata upgrade: use one of the following commands, depending on whether Kerberos is enabled (see Configuring Hadoop Security in CDH 5).

   Important:

   In an HDFS HA deployment, make sure that both the NameNodes and all of the JournalNodes are up and functioning normally before you proceed.

- If Kerberos is enabled:

```
$ kinit -kt /path/to/hdfs.keytab hdfs/<fully.qualified.domain.name@YOUR-REALM.COM>
 && hdfs dfsadmin -finalizeUpgrade
```

- If Kerberos is not enabled:

```
$ sudo -u hdfs hdfs dfsadmin -finalizeUpgrade
```

Note:

After the metadata upgrade completes, the `previous/` and `blocksBeingWritten/` directories in the DataNodes' data directories aren't cleared until the DataNodes are restarted.

Upgrading from CDH 5.0.0 or Later to the Latest Release

Note: Are you on the right page?

Use the instructions on this page only to upgrade from CDH 5.0.0 or later.

To upgrade from a release earlier than CDH 5.0.0:

- Use these instructions to upgrade from a CDH 5 Beta release;
- Use these instructions to upgrade from a CDH 4 release.

Step 1: Prepare the cluster for the upgrade

1. Put the NameNode into safe mode and save the`fsimage`:
   a. Put the NameNode (or active NameNode in an HA configuration) into safe mode:

   ```
   $ sudo -u hdfs hdfs dfsadmin -safemode enter
   ```

   b. Perform a `saveNamespace` operation:

   ```
   $ sudo -u hdfs hdfs dfsadmin -saveNamespace
   ```

   This will result in a new `fsimage` being written out with no edit log entries.

   c. With the NameNode still in safe mode, shut down all services as instructed below.
2. Shut down Hadoop services across your entire cluster by running the following command on every host in your cluster:

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

3. 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
```

Important:

When you are sure that all Hadoop services have been shut down, do the following step. **It is particularly important that the NameNode service is not running so that you can make a consistent backup**.

4. Back up the HDFS metadata on the NameNode machine, as follows.
5.

Note:

- Cloudera recommends backing up HDFS metadata on a regular basis, as well as before a major upgrade.
- `dfs.name.dir` is deprecated but still works; `dfs.namenode.name.dir` is preferred. This example uses `dfs.name.dir`.

a. Find the location of your `dfs.name.dir` (or `dfs.namenode.name.dir`); for example:

```
$ grep -C1 dfs.name.dir /etc/hadoop/conf/hdfs-site.xml
<property> <name>dfs.name.dir</name> <value>/mnt/hadoop/hdfs/name</value>
</property>
```

b. Back up the directory. The path inside the <value> XML element is the path to your HDFS metadata. If you see a comma-separated list of paths, there is no need to back up all of them; they store the same data. Back up the first directory, for example, by using the following commands:

```
$ cd /mnt/hadoop/hdfs/name
# tar -cvf /root/nn_backup_data.tar .
./
./current/
./current/fsimage
./current/fstime
./current/VERSION
./current/edits
./image/
./image/fsimage
```

Warning:

If you see a file containing the word *lock*, the NameNode is probably still running. Repeat the preceding steps from the beginning; start at Step 1 and shut down the Hadoop services.

Step 2: If necessary, download the CDH 5 "1-click"package on each of the hosts in your cluster

Before you begin: Check whether you have the CDH 5 "1-click" repository installed.

- On Red Hat/CentOS-compatible and SLES systems:

```
rpm -q CDH 5-repository
```

If you are upgrading from CDH 5 Beta 1 or later, you should see:

```
CDH5-repository-1-0
```

In this case, skip to Step 3. If instead you see:

```
package CDH 5-repository is not installed
```

proceed with these RHEL instructions or these SLES instructions.

- On Ubuntu and Debian systems:

```
dpkg -l | grep CDH 5-repository
```

If the repository is installed, skip to Step 3; otherwise proceed with these instructions.

Summary: If the CDH 5 "1-click" repository is not already installed on each host in the cluster, follow the instructions below for that host's operating system:

- Instructions for Red Hat-compatible systems
- Instructions for SLES systems
- Instructions for Ubuntu and Debian systems

On Red Hat-compatible systems:

1. Download the CDH 5 "1-click Install" package.

   Click the entry in the table below that matches your Red Hat or CentOS system, choose **Save File**, and save the file to a directory to which you have write access (it can be your home directory).

OS Version	Click this Link
Red Hat/CentOS/Oracle 5	Red Hat/CentOS/Oracle 5 link
Red Hat/CentOS/Oracle 6	Red Hat/CentOS/Oracle 6 link

2. Install the RPM:

   - **Red Hat/CentOS/Oracle 5**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

   - **Red Hat/CentOS/Oracle 6**

   ```
   $ sudo yum --nogpgcheck localinstall cloudera-cdh-5-0.x86_64.rpm
   ```

Note:

For instructions on how to add a CDH 5 yum repository or build your own CDH 5 yum repository, see Installing CDH 5 On Red Hat-compatible systems.

Note: Make sure your repositories are up to date

Before proceeding, make sure the repositories on each system are up to date:

```
sudo yum clean all
```

This ensures that the system repositories contain the latest software (it does not actually install anything).

On SLES systems:

1. Download the CDH 5 "1-click Install" package.

   Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).

2. Install the RPM:

   ```
   $ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
   ```

3. Update your system package index by running:

   ```
   $ sudo zypper refresh
   ```

```
$ sudo rpm -i cloudera-cdh-5-0.x86_64.rpm
```

> **Note:**
>
> For instructions on how to add a repository or build your own repository, see Installing CDH 5 on SLES Systems.

> **Note: Make sure your repositories are up to date**
>
> Before proceeding, make sure the repositories on each system are up to date:
>
> ```
> sudo zypper clean --all
> ```
>
> This ensures that the system repositories contain the latest software (it does not actually install anything).

On Ubuntu and Debian systems:

1. Download the CDH 5 "1-click Install" package:

OS Version	Click this Link
Wheezy	Wheezy link
Precise	Precise link

2. Install the package. Do one of the following:
   - Choose **Open with** in the download window to use the package manager.
   - Choose **Save File**, save the package to a directory to which you have write access (it can be your home directory) and install it from the command line, for example:

     ```
     sudo dpkg -i cdh5-repository_1.0_all.deb
     ```

> **Note:**
>
> For instructions on how to add a repository or build your own repository, see Installing CDH 5 on Ubuntu Systems.

> **Note: Make sure your repositories are up to date**
>
> Before proceeding, make sure the repositories on each system are up to date:
>
> ```
> sudo apt-get update
> ```
>
> This ensures that the system repositories contain the latest software (it does not actually install anything).

Step 3: Upgrade the Packages on the Appropriate Hosts

Upgrade MRv1, YARN, or both, depending on what you intend to use.

Note:

- Remember that you can install and configure both MRv1 and YARN, but you should not run them both on the same set of nodes at the same time.
- If you are using HA for the NameNode, do not install `hadoop-hdfs-secondarynamenode`.

Before installing MRv1 or YARN: (Optionally) add a repository key on each system in the cluster, if you have not already done so. Add the Cloudera Public GPG Key to your repository by executing one of the following commands:

- **For Red Hat/CentOS/Oracle 5 systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Red Hat/CentOS/Oracle 6 systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For all SLES systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Ubuntu Precise systems:**

```
$ curl -s
http://archive.cloudera.com/cdh5/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Debian Wheezy systems:**

```
$ curl -s
http://archive.cloudera.com/cdh5/debian/wheezy/amd64/cdh/archive.key
| sudo apt-key add -
```

Step 3a: If you are using MRv1, upgrade the MRv1 packages on the appropriate hosts.

Skip this step if you are using YARN exclusively. Otherwise upgrade each type of daemon package on the appropriate hosts as follows:

1. Install and deploy ZooKeeper:

   Important:

   Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
JobTracker host running:	

Where to install	Install commands
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-jobtracker`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-jobtracker`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-jobtracker`
NameNode host running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the JobTracker, NameNode, and Secondary (or Standby) NameNode hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode`
All client hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-client`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-client`

Step 3b: If you are using YARN, upgrade the YARN packages on the appropriate hosts.

Skip this step if you are using MRv1 exclusively. Otherwise upgrade each type of daemon package on the appropriate hosts as follows:

1. Install and deploy ZooKeeper:

> **Important:**
>
> Cloudera recommends that you install (or update) and start a ZooKeeper cluster before proceeding. This is a **requirement** if you are deploying high availability (HA) for the NameNode or JobTracker.

   Follow instructions under ZooKeeper Installation.

2. Install each type of daemon package on the appropriate systems(s), as follows.

Where to install	Install commands
Resource Manager host (analogous to MRv1 JobTracker) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager`
NameNode host running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-namenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode`
Secondary NameNode host (if used) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode`
All cluster hosts except the Resource Manager (analogous to MRv1 TaskTrackers) running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`

Where to install	Install commands
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce`
One host in the cluster running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
Ubuntu or Debian	`$ sudo apt-get update; sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver`
All client hosts, running:	
Red Hat/CentOS compatible	`$ sudo yum clean all; sudo yum install hadoop-client`
SLES	`$ sudo zypper clean --all; sudo zypper install hadoop-client`
Ubuntu or Debian	`sudo apt-get update; sudo apt-get install hadoop-client`

Note:

The `hadoop-yarn` and `hadoop-hdfs` packages are installed on each system automatically as dependencies of the other packages.

Step 4: In an HA Deployment, Upgrade and Start the Journal Nodes

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 proceeding to the next step.

> **Important:**
>
> In an HA deployment, the JournalNodes must be up and running CDH 5 before you proceed.

Step 5: Start HDFS

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

Step 6: Start MapReduce (MRv1) or YARN

You are now ready to start and test MRv1 or YARN.

For MRv1	For YARN
Start MRv1	Start YARN and the MapReduce JobHistory Server

Step 6a: Start MapReduce (MRv1)

> **Important:**
>
> Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 6a and 6b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start MapReduce. On each TaskTracker system:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

On the JobTracker system:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

Verify that the JobTracker and TaskTracker started properly.

```
$ sudo jps | grep Tracker
```

If the permissions of directories are not configured correctly, the JobTracker and TaskTracker processes start and immediately fail. If this happens, check the JobTracker and TaskTracker logs and set the permissions correctly.

Verify basic cluster operation for MRv1.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

> **Note:**
>
> For important configuration information, see Deploying MapReduce v1 (MRv1) on a Cluster.

1. Create a home directory on HDFS for the user who will be running the job (for example, `joe`):

   ```
   sudo -u hdfs hadoop fs -mkdir -p /user/joe sudo -u hdfs hadoop fs -chown joe
   /user/joe
   ```

   Do the following steps as the user `joe`.

2. Make a directory in HDFS called `input` and copy some XML files into it by running the following commands:

   ```
   $ hadoop fs -mkdir input
   $ hadoop fs -put /etc/hadoop/conf/*.xml input
   $ hadoop fs -ls input
   Found 3 items:
   -rw-r--r-- 1 joe supergroup 1348 2012-02-13 12:21 input/core-site.xml
   -rw-r--r-- 1 joe supergroup 1913 2012-02-13 12:21 input/hdfs-site.xml
   -rw-r--r-- 1 joe supergroup 1001 2012-02-13 12:21 input/mapred-site.xml
   ```

3. Run an example Hadoop job to grep with a regular expression in your input data.

   ```
   $ /usr/bin/hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar grep input
    output 'dfs[a-z.]+'
   ```

4. After the job completes, you can find the output in the HDFS directory named `output` because you specified that output directory to Hadoop.

   ```
   $ hadoop fs -ls
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:36 /user/joe/input
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:38 /user/joe/output
   ```

   You can see that there is a new directory called `output`.

5. List the output files.

   ```
   $ hadoop fs -ls output
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-02-25 10:33 /user/joe/output/_logs
   -rw-r--r-- 1 joe supergroup 1068 2009-02-25 10:33 /user/joe/output/part-00000
   -rw-r--r- 1 joe supergroup 0 2009-02-25 10:33 /user/joe/output/_SUCCESS
   ```

6. Read the results in the output file; for example:

   ```
   $ hadoop fs -cat output/part-00000 | head
   1 dfs.datanode.data.dir
   1 dfs.namenode.checkpoint.dir
   1 dfs.namenode.name.dir
   1 dfs.replication
   1 dfs.safemode.extension
   1 dfs.safemode.min.datanodes
   ```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Step 6b: Start MapReduce with YARN

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade your performance and may result in an unstable MapReduce cluster deployment. Steps 6a and 6b are mutually exclusive.

After you have verified HDFS is operating correctly, you are ready to start YARN. First, if you have not already done so, create directories and set the correct permissions.

Note: For more information see Deploying MapReduce v2 (YARN) on a Cluster.

Create a history directory and set permissions; for example:

```
$ sudo -u hdfs hadoop fs -mkdir -p /user/history
$ sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
 $ sudo -u hdfs hadoop fs -chown yarn /user/history
```

Create the `/var/log/hadoop-yarn` directory and set ownership:

```
$ sudo -u hdfs hadoop fs -mkdir -p /var/log/hadoop-yarn
$ sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn
```

Note: You need to create this directory because it is the parent of `/var/log/hadoop-yarn/apps` which is explicitly configured in the `yarn-site.xml`.

Verify the directory structure, ownership, and permissions:

```
$ sudo -u hdfs hadoop fs -ls -R /
```

You should see:

```
drwxrwxrwt - hdfs supergroup 0 2012-04-19 14:31 /tmp
drwxr-xr-x - hdfs supergroup 0 2012-05-31 10:26 /user
drwxrwxrwt - yarn supergroup 0 2012-04-19 14:31 /user/history
drwxr-xr-x - hdfs supergroup 0 2012-05-31 15:31 /var
drwxr-xr-x - hdfs supergroup 0 2012-05-31 15:31 /var/log
drwxr-xr-x - yarn mapred 0 2012-05-31 15:31 /var/log/hadoop-yarn
```

To start YARN, start the ResourceManager and NodeManager services:

Note:

Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

```
$ sudo service hadoop-yarn-resourcemanager start
```

On each NodeManager system (typically the same ones where DataNode service runs):

```
$ sudo service hadoop-yarn-nodemanager start
```

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

```
$ sudo service hadoop-mapreduce-historyserver start
```

For each user who will be submitting MapReduce jobs using MapReduce v2 (YARN), or running Pig, Hive, or Sqoop 1 in a YARN installation, set the `HADOOP_MAPRED_HOME` environment variable as follows:

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

Verify basic cluster operation for YARN.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running an example from the Apache Hadoop web site.

Note:

For important configuration information, see Deploying MapReduce v2 (YARN) on a Cluster.

1. Create a home directory on HDFS for the user who will be running the job (for example, `joe`):

   ```
   $ sudo -u hdfs hadoop fs -mkdir -p /user/joe sudo -u hdfs hadoop fs -chown joe
   /user/joe
   ```

   Do the following steps as the user `joe`.

2. Make a directory in HDFS called `input` and copy some XML files into it by running the following commands in pseudo-distributed mode:

   ```
   $ hadoop fs -mkdir input
   $ hadoop fs -put /etc/hadoop/conf/*.xml input
   $ hadoop fs -ls input
   Found 3 items:
   -rw-r--r-- 1 joe supergroup 1348 2012-02-13 12:21 input/core-site.xml
   -rw-r--r-- 1 joe supergroup 1913 2012-02-13 12:21 input/hdfs-site.xml
   -rw-r--r-- 1 joe supergroup 1001 2012-02-13 12:21 input/mapred-site.xml
   ```

3. Set `HADOOP_MAPRED_HOME` for user `joe`:

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

4. Run an example Hadoop job to `grep` with a regular expression in your input data.

   ```
   $ hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar grep input
   output23 'dfs[a-z.]+'
   ```

5. After the job completes, you can find the output in the HDFS directory named `output23` because you specified that output directory to Hadoop.

   ```
   $ hadoop fs -ls
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:36 /user/joe/input
   drwxr-xr-x - joe supergroup 0 2009-08-18 18:38 /user/joe/output23
   ```

   You can see that there is a new directory called `output23`.

6. List the output files:

   ```
   $ hadoop fs -ls output23
   Found 2 items
   drwxr-xr-x - joe supergroup 0 2009-02-25 10:33 /user/joe/output23/_SUCCESS
   -rw-r--r-- 1 joe supergroup 1068 2009-02-25 10:33 /user/joe/output23/part-r-00000
   ```

7. Read the results in the output file:

   ```
   $ hadoop fs -cat output23/part-r-00000 | head
   1 dfs.safemode.min.datanodes
   ```

```
1 dfs.safemode.extension
1 dfs.replication
1 dfs.permissions.enabled
1 dfs.namenode.name.dir
1 dfs.namenode.checkpoint.dir
1 dfs.datanode.data.dir
```

You have now confirmed your cluster is successfully running CDH 5.

Important:

If you have client hosts, make sure you also update them to CDH 5, and upgrade the components running on those clients as well.

Step 7: Set the Sticky Bit

For security reasons Cloudera strongly recommends you set the sticky bit on directories if you have not already done so.

The sticky bit prevents anyone except the superuser, directory owner, or file owner from deleting or moving the files within a directory. (Setting the sticky bit for a file has no effect.) Do this for directories such as `/tmp`. (For instructions on creating `/tmp` and setting its permissions, see these instructions).

Step 8: Upgrade Components

Note:

- For important information on new and changed components, see the Release Notes. To see whether there is a new version of a particular component in CDH 5, check the CDH Version and Packaging Information.
- Cloudera recommends that you regularly update the software on each system in the cluster (for example, on a RHEL-compatible system, regularly run `yum update`) to ensure that all the dependencies for any given component are up to date. (If you have not been in the habit of doing this, be aware that the command may take a while to run the first time you use it.)

CDH 5 Components

Use the following sections to install or upgrade CDH 5 components:

- Crunch Installation on page 155
- Flume Installation on page 157
- HBase Installation on page 169
- Installing and Using HCatalog on page 203
- Hive Installation on page 233
- HttpFS Installation on page 259
- Hue Installation on page 263
- Impala Installation on page 211
- Llama Installation on page 291
- Mahout Installation on page 293
- Oozie Installation on page 297
- Pig Installation on page 319
- Search Installation on page 325
- Sentry Installation on page 327
- Snappy Installation on page 329
- Spark Installation on page 333

- Sqoop 1 Installation on page 339
- Sqoop 2 Installation on page 345
- Whirr Installation on page 353
- ZooKeeper Installation

See also the instructions for installing or updating LZO.

Step 9: Apply Configuration File Changes if Necessary

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

For example, if you have modified your `zoo.cfg` configuration file (`/etc/zookeeper/zoo.cfg`), the upgrade renames and preserves a copy of your modified `zoo.cfg` as `/etc/zookeeper/zoo.cfg.rpmsave`. If you have not already done so, you should now compare this to the new `/etc/zookeeper/conf/zoo.cfg`, resolve differences, and make any changes that should be carried forward (typically where you have changed property value defaults). Do this for each component you upgrade.

Migrating data between a CDH 4 and CDH 5 cluster

You can migrate the data from a CDH 4 (or any Apache 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. This can be useful if you are not planning to upgrade your CDH 4 cluster itself at this point. The following sections provide information and instructions:

- Requirements
- Using DistCp to Migrate Data between two Clusters
- Post-Migration Verification

Requirements and Restrictions

1. The CDH 5 cluster must have a MapReduce service running on it (MRv1 or YARN (MRv2)).
2. All the MapReduce nodes in the CDH 5 cluster should have full network access to all the nodes of the source cluster. This allows you to perform the copy in a distributed manner.
3. To copy data between a secure and an insecure cluster, you must run the `distcp` command on the secure cluster.
4. To copy data from a CDH 4 to a CDH 5 cluster, you can do one of the following:

Note:

The term **source** in this case refers to the CDH 4 (or other Hadoop) cluster you want to migrate or copy data from; and **destination** refers to the CDH 5 cluster.

- Running commands on the destination cluster, use the Hftp protocol for the source cluster, and HDFS for the destination. (Hftp is read-only, so you must run DistCp on the destination cluster and pull the data from the source cluster.) See Copying Data between two Clusters Using DistCp and Hftp on page 106.

  Note:

  Do not use this method if one of the clusters is secure and the other is not.

- Running commands on the source cluster, use the HDFS or webHDFS protocol for the source cluster, and webHDFS for the destination. See Copying Data between a Secure and an Insecure Cluster using DistCp and webHDFS on page 106.
- Running commands on the destination cluster, use webHDFS for the source cluster, and webHDFS for the destination. See Copying Data between a Secure and an Insecure Cluster using DistCp and webHDFS on page 106.

The following restrictions currently apply (see Apache Hadoop Known Issues):

- DistCp does not work between a secure cluster and an insecure cluster in some cases.

  As of CDH 5.1.3, DistCp *does* work between a secure and an insecure cluster if you use the webHDFS protocol and run the command from the secure cluster side after setting `ipc.client.fallback-to-simple-auth-allowed` to true, as described under Copying Data between a Secure and an Insecure Cluster using DistCp and webHDFS on page 106.

- To use DistCp using Hftp from a secure cluster using SPNEGO, you must configure the `dfs.https.port` property on the client to use the HTTP port (50070 by default).

Copying Data between two Clusters Using DistCp and Hftp

You can use the DistCp tool on the CDH 5 cluster to initiate the copy job to move the data. Between two clusters running different versions of CDH, run the DistCp tool with `hftp://` as the source file system and `hdfs://` as the destination file system. This uses the HFTP protocol for the source, and the HDFS protocol for the destination. the default port for HFTP is 50070, and the default port for HDFS is 8020.

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

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

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

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

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

The DistCp Command

For more help, and to see all the options available on the DistCp tool, use the following command to see the built-in help:

```
$ hadoop distcp
```

Run the DistCp copy by issuing a command such as the following on the CDH 5 cluster:

- **Important:** Run the following DistCp commands on the destination cluster only, in this example, the CDH 5 cluster.

```
$ hadoop distcp hftp://cdh4-namenode:50070/ hdfs://CDH5-nameservice/
```

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

```
$ hadoop distcp hftp://cdh4-namenode:50070/hbase hdfs://CDH5-nameservice/hbase
```

DistCp will then submit a regular MapReduce job that performs a file-by-file copy.

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

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

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

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

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

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

Post-migration Verification

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

Configuring Ports for CDH 5

The CDH 5 components, and third parties such as Kerberos, use the ports listed in the tables that follow. Before you deploy CDH 5, make sure these ports are open on each system. If you cannot follow the recommendations exactly, you must completely disable `iptables`.

- Ports Used by Components of CDH 5 on page 109
- Ports Used by Third-Party Components on page 113

Ports Used by Components of CDH 5

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
Hadoop HDFS	DataNode		50010	TCP	External	`dfs.datanode.address`	DataNode HTTP server port
	DataNode	Secure	1004	TCP	External	`dfs.datanode.address`	
	DataNode		50075	TCP	External	`dfs.datanode.http.address`	
	DataNode	Secure	1006	TCP	External	`dfs.datanode.http.address`	
	DataNode		50020	TCP	External	`dfs.datanode.ipc.address`	
	NameNode		8020	TCP	External	`fs.default.name` or `fs.defaultFS`	`fs.default.name` is deprecated (but still works)
	NameNode		50070	TCP	External	`dfs.http.address` or `dfs.namenode.http-address`	`dfs.http.address` is deprecated (but still works)
	NameNode	Secure	50470	TCP	External	`dfs.https.address` or `dfs.namenode.https-address`	`dfs.https.address` is deprecated (but still works)
	Secondary NameNode		50090	TCP	Internal	`dfs.secondary.http.address` or `dfs.namenode.secondary.http-address`	`dfs.secondary.http.address` is deprecated (but still works)
	Secondary NameNode	Secure	50495	TCP	Internal	`dfs.secondary.https.address`	
	JournalNode		8485	TCP	Internal	`dfs.namenode.shared.edits.dir`	

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	JournalNode		8480	TCP	Internal		
Hadoop MapReduce (MRv1)	JobTracker		8021	TCP	External	`mapred.job.tracker`	
	JobTracker		50030	TCP	External	`mapred.job.tracker.http.address`	
	JobTracker	Thrift Plugin	9290	TCP	Internal	`jobtracker.thrift.address`	Required by Hue and Cloudera Manager Activity Monitor
	TaskTracker		50060	TCP	External	`mapred.task.tracker.http.address`	
	TaskTracker		0	TCP	Localhost	`mapred.task.tracker.report.address`	Communicating with child (umbilical)
Hadoop YARN (MRv2)	ResourceManager		8032	TCP	External	`yarn.resourcemanager.address`	
	ResourceManager		8030	TCP	Internal	`yarn.resourcemanager.scheduler.address`	
	ResourceManager		8031	TCP	Internal	`yarn.resourcemanager.resource-tracker.address`	
	ResourceManager		8033	TCP	External	`yarn.resourcemanager.admin.address`	
	ResourceManager		8088	TCP	External	`yarn.resourcemanager.webapp.address`	
	NodeManager		8040	TCP	Internal	`yarn.nodemanager.localizer.address`	
	NodeManager		8042	TCP	External	`yarn.nodemanager.webapp.address`	
	NodeManager		8041	TCP	Internal	`yarn.nodemanager.address`	
	MapReduce JobHistory Server		10020	TCP	Internal	`mapreduce.jobhistory.address`	
	Shuffle HTTP		13562	TCP	Internal		
	MapReduce JobHistory Server		19888	TCP	External	`mapreduce.jobhistory.webapp.address`	

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
HBase	Master		60000	TCP	External	`hbase.master.port`	IPC
	Master		60010	TCP	External	`hbase.master.info.port`	HTTP
	RegionServer		60020	TCP	External	`hbase.regionserver.port`	IPC
	RegionServer		60030	TCP	External	`hbase.regionserver.info.port`	HTTP
	HQuorumPeer		2181	TCP		`hbase.zookeeper.property.clientPort`	HBase-managed ZK mode
	HQuorumPeer		2888	TCP		`hbase.zookeeper.peerport`	HBase-managed ZK mode
	HQuorumPeer		3888	TCP		`hbase.zookeeper.leaderport`	HBase-managed ZK mode
	REST	REST Service	8080	TCP	External	`hbase.rest.port`	
	REST UI		8085	TCP	External		
	ThriftServer	Thrift Server	9090	TCP	External	Pass `-p <port>` on CLI	
	ThriftServer		9095	TCP	External		
		Avro server	9090	TCP	External	Pass `--port <port>` on CLI	
Hive	Metastore		9083	TCP	External		
	HiveServer2		10000	TCP	External	`hive.server2.thrift.port`	
Sqoop	Metastore		16000	TCP	External	`sqoop.metastore.server.port`	
Sqoop 2	Sqoop 2 server		12000	TCP	External		
	Sqoop 2		12001	TCP	External		Admin port
ZooKeeper	Server (with CDH 5 and/or Cloudera Manager 5)		2181	TCP	External	clientPort	Client port
	Server (with CDH 5 only)		2888	TCP	Internal	`X in server.N =host:X:Y`	Peer

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	Server (with CDH 5 only)		3888	TCP	Internal	`X in server.N =host:X:Y`	Peer
	Server (with CDH 5 and Cloudera Manager 5)		3181	TCP	Internal	`X in server.N =host:X:Y`	Peer
	Server (with CDH 5 and Cloudera Manager 5)		4181	TCP	Internal	`X in server.N =host:X:Y`	Peer
	ZooKeeper FailoverController (ZKFC)		8019	TCP	Internal		Used for HA
	ZooKeeper JMX port		9010	TCP	Internal		ZooKeeper will also use another randomly selected port for RMI. To allow Cloudera Manager to monitor ZooKeeper, you must *EITHER* ▪ Open up all ports when the connection originates from the Cloudera Manager server; *OR* ▪ Do the following: **1.** Open a non-ephemeral port (such as 9011) in the firewall. **2.** Install Oracle Java 7u4 JDK or later. **3.** Add the port configuration to the safety valve, for example: `Dcom.sun.management.jmxremote.port=9011` **4.** Restart ZooKeeper.
Hue	Server		8888	TCP	External		
	Beeswax Server		8002		Internal		

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	Beeswax Metastore		8003		Internal		
Oozie	Oozie Server		11000	TCP	External	`OOZIE_HTTP_PORT` in `oozie-env.sh`	HTTP
	Oozie Server		11001	TCP	localhost	`OOZIE_ADMIN_PORT` in `oozie-env.sh`	Shutdown port
Spark	Default Master RPC port		7077	TCP	External		
	Default Worker RPC port		7078	TCP			
	Default Master web UI port		18080	TCP	External		
	Default Worker web UI port		18081	TCP			
HttpFS	HttpFS		14000	TCP			
	HttpFS		14001	TCP			

Ports Used by Third-Party Components

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
Ganglia	ganglia-gmond		8649	UDP/TCP	Internal		
	ganglia-web		80	TCP	External	Via Apache `httpd`	
Kerberos	KRB5 KDC Server	Secure	88	UDP/TCP	External	`kdc_ports` and `kdc_tcp_ports` in either the `[kdcdefaults]` or `[realms]` sections of	By default only UDP

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
						`kdc.conf`	
	KRB5 Admin Server	Secure	749	TCP	Internal	`kadmind_port` in the `[realms]` section of `kdc.conf`	

Deploying CDH 5 in Pseudo-Distributed Mode

In pseudo-distributed mode, Hadoop processing is distributed over all of the cores/processors on a single machine. Hadoop writes all files to the Hadoop Distributed FileSystem (HDFS), and all services and daemons communicate over local TCP sockets for inter-process communication.

To deploy CDH 5 in pseudo-distributed mode (and to install it if you have not already done so) follow the instructions in the CDH 5 Quick Start Guide.

Deploying CDH 5 on a Cluster

Note: Do the tasks in this section after installing the latest version of CDH; see CDH 5 Installation.

To deploy CDH 5 on a cluster, do the following:

1. Configure Network Hosts
2. Configure HDFS
3. Deploy YARN with MapReduce v2 (YARN) *or* MapReduce v1 (MRv1)
4. Also see, Centralized Cache Management in HDFS on page 147

See Hadoop Users in CDH 5 for a list of special users created by CDH 5 during installation.

Configuring Network Names

To ensure that the members of the cluster can communicate with each other, do the following on every system.

Important:

CDH requires IPv4. IPv6 is not supported.

1. Set the hostname of each system to a unique name (not `localhost`). For example:

```
$ sudo hostname myhost-1
```

Note: This is a temporary measure only. The hostname set by `hostname` does not survive across reboots

2. Make sure the `/etc/hosts` file on each system contains the IP addresses and fully-qualified domain names (FQDN) of all the members of the cluster.

Important:

The canonical name of each host in `/etc/hosts` **must** be the FQDN (for example `myhost-1.mynet.myco.com`), not the unqualified hostname (for example `myhost-1`). The canonical name is the first entry after the IP address.

If you are using DNS, storing this information in `/etc/hosts` is not required, but it is good practice.

3. Make sure the `/etc/sysconfig/network` file on each system contains the hostname you have just set (or verified) for that system, for example `myhost-1`.
4. Check that this system is consistently identified to the network:

   a. Run `uname -a` and check that the hostname matches the output of the `hostname` command.
   b. Run `/sbin/ifconfig` and note the value of `inet addr` in the `eth0` entry, for example:

```
$ /sbin/ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0C:29:A4:E8:97
          inet addr:172.29.82.176  Bcast:172.29.87.255  Mask:255.255.248.0
...
```

c. Run `host -v -t A `hostname`` and make sure that hostname matches the output of the `hostname` command, and has the same IP address as reported by `ifconfig` for `eth0`; for example:

```
$ host -v -t A `hostname`
Trying "myhost.mynet.myco.com"
...
;; ANSWER SECTION:
myhost.mynet.myco.com. 60 IN A 172.29.82.176
```

5. For MRv1: make sure `conf/core-site.xml` and `conf/mapred-site.xml`, respectively, have the **hostnames** – not the IP addresses – of the NameNode and the JobTracker. These can be FQDNs (for example `myhost-1.mynet.myco.com`), or unqualified hostnames (for example `myhost-1`). See Customizing Configuration Files and Deploying MapReduce v1 (MRv1) on a Cluster.
6. For YARN: make sure `conf/core-site.xml` and `conf/yarn-site.xml`, respectively, have the **hostnames** – not the IP addresses – of the NameNode, the ResourceManager, and the ResourceManager Scheduler. See Customizing Configuration Files and Deploying MapReduce v2 (YARN) on a Cluster.
7. Make sure that components that depend on a client-server relationship – Oozie, HBase, ZooKeeper – are configured according to the instructions on their installation pages:
   - Oozie Installation
   - HBase Installation
   - ZooKeeper Installation

Deploying HDFS on a Cluster

Important:

For instructions for configuring High Availability (HA) for the NameNode, see the CDH 5 High Availability Guide. For instructions on using HDFS Access Control Lists (ACLs), see Enabling HDFS Extended ACLs.

Proceed as follows to deploy HDFS on a cluster. Do this for all clusters, whether you are deploying MRv1 or YARN:

1. Copy the Hadoop configuration
2. Customize configuration files
3. Configure Local Storage Directories
4. Configure DataNodes to tolerate local storage directory failure
5. Format the NameNode
6. Configure a remote NameNode storage directory
7. Configure the Secondary NameNode (if used)
8. Optionally enable Trash
9. Optionally configure DataNode storage balancing
10. Optionally enable WebHDFS
11. Optionally configure LZO
12. Start HDFS on page 128
13. Deploy MRv1 or YARN and start services

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in`/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Copying the Hadoop Configuration and Setting Alternatives

To customize the Hadoop configuration:

1. Copy the default configuration to your custom directory:

```
$ sudo cp -r /etc/hadoop/conf.empty /etc/hadoop/conf.my_cluster
```

   You can call this configuration anything you like; in this example, it's called `my_cluster`.

   Important:

   When performing the configuration tasks in this section, and when you go on to deploy MRv1 or YARN, edit the configuration files in this custom directory. Do not create your custom configuration in the default directory `/etc/hadoop/conf.empty`.

2. CDH uses the `alternatives` setting to determine which Hadoop configuration to use. Set `alternatives` to point to your custom directory, as follows.

   To manually set the configuration on Red Hat-compatible systems:

```
$ sudo alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50
$ sudo alternatives --set hadoop-conf /etc/hadoop/conf.my_cluster
```

   To manually set the configuration on Ubuntu and SLES systems:

```
$ sudo update-alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50
$ sudo update-alternatives --set hadoop-conf /etc/hadoop/conf.my_cluster
```

   This tells CDH to use the configuration in `/etc/hadoop/conf.my_cluster`.

You can display the current `alternatives` setting as follows.

To display the current setting on Red Hat-compatible systems:

```
sudo alternatives --display hadoop-conf
```

To display the current setting on Ubuntu, Debian, and SLES systems:

```
sudo update-alternatives --display hadoop-conf
```

You should see output such as the following:

```
hadoop-conf - status is auto.
link currently points to /etc/hadoop/conf.my_cluster
/etc/hadoop/conf.my_cluster - priority 50
/etc/hadoop/conf.empty - priority 10
Current `best' version is /etc/hadoop/conf.my_cluster.
```

Because the configuration in `/etc/hadoop/conf.my_cluster` has the highest priority (50), that is the one CDH will use. For more information on alternatives, see the `update-alternatives(8)` man page on Ubuntu and SLES systems or the `alternatives(8)` man page On Red Hat-compatible systems.

Customizing Configuration Files

The following tables show the most important properties that you must configure for your cluster.

Note:

For information on other important configuration properties, and the configuration files, see the Apache Cluster Setup page.

Property	Configuration File	Description
`fs.defaultFS`	`core-site.xml`	Note: `fs.default.name` is deprecated. Specifies the NameNode and the default file system, in the form `hdfs://<namenode host>:<namenode port>/`. The default value is `file///`. The default file system is used to resolve relative paths; for example, if `fs.default.name` or `fs.defaultFS` is set to `hdfs://mynamenode/`, the relative URI `/mydir/myfile` resolves to `hdfs://mynamenode/mydir/myfile`. Note: for the cluster to function correctly, the `<namenode>` part of the string **must** be the hostname (for example `mynamenode`), or the HA-enabled logical URI, not the IP address.
`dfs.permissions.superusergroup`	`hdfs-site.xml`	Specifies the UNIX group containing users that will be treated as superusers by HDFS. You can stick with the value of 'hadoop' or pick your own group depending on the security policies at your site.

Sample Configuration

core-site.xml:

```
<property>
 <name>fs.defaultFS</name>
 <value>hdfs://namenode-host.company.com:8020</value>
</property>
```

hdfs-site.xml:

```
<property>
 <name>dfs.permissions.superusergroup</name>
 <value>hadoop</value>
</property>
```

Configuring Local Storage Directories

You need to specify, create, and assign the correct permissions to the local directories where you want the HDFS daemons to store data. You specify the directories by configuring the following two properties in the `hdfs-site.xml` file.

Property	Configuration File Location	Description
`dfs.name.dir` or `dfs.namenode.name.dir`	`hdfs-site.xml` on the NameNode	This property specifies the URIs of the directories where the NameNode stores its metadata and edit logs. Cloudera recommends that you specify at least two directories. One of these should be located on an NFS mount point, unless you will be using a High Availability (HA) configuration.
`dfs.data.dir` or `dfs.datanode.data.dir`	`hdfs-site.xml` on each DataNode	This property specifies the URIs of the directories where the DataNode stores blocks. Cloudera recommends that you configure the disks on the DataNode in a JBOD configuration, mounted at `/data/1/` through `/data/N`, and configure `dfs.data.dir` or `dfs.datanode.data.dir` to specify `file:///data/1/dfs/dn` through `file:///data/N/dfs/dn/`.

Note:

`dfs.data.dir` and `dfs.name.dir` are deprecated; you should use `dfs.datanode.data.dir` and `dfs.namenode.name.dir` instead, though `dfs.data.dir` and `dfs.name.dir` will still work.

Sample configuration:

hdfs-site.xml on the NameNode:

```
<property>
 <name>dfs.namenode.name.dir</name>
 <value>file:///data/1/dfs/nn,file:///nfsmount/dfs/nn</value>
</property>
```

hdfs-site.xml on each DataNode:

```
<property>
 <name>dfs.datanode.data.dir</name>

<value>file:///data/1/dfs/dn,file:///data/2/dfs/dn,file:///data/3/dfs/dn,file:///data/4/dfs/dn</value>
</property>
```

After specifying these directories as shown above, you must create the directories and assign the correct file permissions to them on each node in your cluster.

In the following instructions, local path examples are used to represent Hadoop parameters. Change the path examples to match your configuration.

Local directories:

- The `dfs.name.dir` or `dfs.namenode.name.dir` parameter is represented by the `/data/1/dfs/nn` and `/nfsmount/dfs/nn` path examples.
- The `dfs.data.dir` or `dfs.datanode.data.dir` parameter is represented by the `/data/1/dfs/dn`, `/data/2/dfs/dn`, `/data/3/dfs/dn`, and `/data/4/dfs/dn` examples.

To configure local storage directories for use by HDFS:

1. On a NameNode host: create the `dfs.name.dir` or `dfs.namenode.name.dir` local directories:

```
$ sudo mkdir -p /data/1/dfs/nn /nfsmount/dfs/nn
```

Important:

If you are using High Availability (HA), you should **not** configure these directories on an NFS mount; configure them on local storage.

2. On all DataNode hosts: create the `dfs.data.dir` or `dfs.datanode.data.dir` local directories:

```
$ sudo mkdir -p /data/1/dfs/dn /data/2/dfs/dn /data/3/dfs/dn /data/4/dfs/dn
```

3. Configure the owner of the `dfs.name.dir` or `dfs.namenode.name.dir` directory, and of the `dfs.data.dir` or `dfs.datanode.data.dir` directory, to be the hdfs user:

```
$ sudo chown -R hdfs:hdfs /data/1/dfs/nn /nfsmount/dfs/nn /data/1/dfs/dn
/data/2/dfs/dn /data/3/dfs/dn /data/4/dfs/dn
```

Here is a summary of the correct owner and permissions of the local directories:

Directory	Owner	Permissions (see Footnote 1)
`dfs.name.dir` or `dfs.namenode.name.dir`	`hdfs:hdfs`	drwx------
`dfs.data.dir` or `dfs.datanode.data.dir`	`hdfs:hdfs`	drwx------

Footnote: 1 The Hadoop daemons automatically set the correct permissions for you on `dfs.data.dir` or `dfs.datanode.data.dir`. But in the case of `dfs.name.dir` or `dfs.namenode.name.dir`, permissions are currently incorrectly set to the file-system default, usually drwxr-xr-x (755). Use the chmod command to reset permissions for these `dfs.name.dir` or `dfs.namenode.name.dir` directories to drwx------ (700); for example:

```
$ sudo chmod 700 /data/1/dfs/nn /nfsmount/dfs/nn
```

or

```
$ sudo chmod go-rx /data/1/dfs/nn /nfsmount/dfs/nn
```

Note:

If you specified nonexistent directories for the `dfs.data.dir` or `dfs.datanode.data.dir` property in the `hdfs-site.xml` file, CDH 5 will shut down. (In previous releases, CDH silently ignored nonexistent directories for `dfs.data.dir`.)

Configuring DataNodes to Tolerate Local Storage Directory Failure

By default, the failure of a single `dfs.data.dir` or `dfs.datanode.data.dir` will cause the HDFS DataNode process to shut down, which results in the NameNode scheduling additional replicas for each block that is present on the DataNode. This causes needless replications of blocks that reside on disks that have not failed.

To prevent this, you can configure DataNodes to tolerate the failure of `dfs.data.dir` or `dfs.datanode.data.dir` directories; use the `dfs.datanode.failed.volumes.tolerated` parameter in `hdfs-site.xml`. For example, if the value for this parameter is 3, the DataNode will only shut down after four or more data directories have failed. This value is respected on DataNode startup; in this example the DataNode will start up as long as no more than three directories have failed.

Note:

It is important that `dfs.datanode.failed.volumes.tolerated` not be configured to tolerate too many directory failures, as the DataNode will perform poorly if it has few functioning data directories.

Formatting the NameNode

Before starting the NameNode for the first time you need to format the file system.

Important:

- Make sure you format the NameNode as user `hdfs`.
- If you are re-formatting the NameNode, keep in mind that this invalidates the DataNode storage locations, so you should remove the data under those locations after the NameNode is formatted.

```
$ sudo -u hdfs hdfs namenode -format
```

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>`

You'll get a confirmation prompt; for example:

```
Re-format filesystem in /data/namedir ? (Y or N)
```

Note: Respond with an **upper-case** `Y`; if you use lower case, the process will abort.

Configuring a Remote NameNode Storage Directory

You should configure the NameNode to write to multiple storage directories, including one remote NFS mount. To keep NameNode processes from hanging when the NFS server is unavailable, configure the NFS mount as a `soft` mount (so that I/O requests that time out fail rather than hang), and set other options as follows:

```
tcp,soft,intr,timeo=10,retrans=10
```

These options configure a soft mount over TCP; transactions will be retried ten times (`retrans=10`) at 1-second intervals (`timeo=10`) before being deemed to have failed.

Example:

```
mount -t nfs -o tcp,soft,intr,timeo=10,retrans=10, <server>:<export> <mount_point>
```

where `<server>` is the remote host, `<export>` is the exported file system, and `<mount_point>` is the local mount point.

Note:

Cloudera recommends similar settings for shared HA mounts, as in the example that follows.

Example for HA:

```
mount -t nfs -o tcp,soft,intr,timeo=50,retrans=12, <server>:<export> <mount_point>
```

Note that in the HA case `timeo` should be set to 50 (five seconds), rather than 10 (1 second), and `retrans` should be set to 12, giving an overall timeout of 60 seconds.

For more information, see the man pages for `mount` and `nfs`.

Configuring Remote Directory Recovery

You can enable the `dfs.namenode.name.dir.restore` option so that the NameNode will attempt to recover a previously failed NameNode storage directory on the next checkpoint. This is useful for restoring a remote storage directory mount that has failed because of a network outage or intermittent NFS failure.

Configuring the Secondary NameNode

Important:

The Secondary NameNode does not provide failover or High Availability (HA). If you intend to configure HA for the NameNode, skip this section: do not install or configure the Secondary NameNode (the Standby NameNode performs checkpointing). After completing the HA software configuration, follow the installation instructions under Deploying HDFS High Availability.

In non-HA deployments, configure a Secondary NameNode that will periodically merge the EditLog with the FSImage, creating a new FSImage which incorporates the changes which were in the EditLog. This reduces the amount of disk space consumed by the EditLog on the NameNode, and also reduces the restart time for the Primary NameNode.

A standard Hadoop cluster (not a Hadoop Federation or HA configuration), can have only one Primary NameNode plus one Secondary NameNode. On production systems, the Secondary NameNode should run on a different machine from the Primary NameNode to improve scalability (because the Secondary NameNode does not compete with the NameNode for memory and other resources to create the system snapshot) and durability (because the copy of the metadata is on a separate machine that is available if the NameNode hardware fails).

Configuring the Secondary NameNode on a Separate Machine

To configure the Secondary NameNode on a separate machine from the NameNode, proceed as follows.

1. Add the name of the machine that will run the Secondary NameNode to the `masters` file.
2. Add the following property to the `hdfs-site.xml` file:

```
<property>
  <name>dfs.namenode.http-address</name>
  <value><namenode.host.address>:50070</value>
  <description>
    The address and the base port on which the dfs NameNode Web UI will listen.
  </description>
</property>
```

Note:

- `dfs.http.address` is deprecated; use `dfs.namenode.http-address`.
- In most cases, you should set `dfs.namenode.http-address` to a routable IP address with port 50070. However, in some cases such as Amazon EC2, when the NameNode should bind to multiple local addresses, you may want to set `dfs.namenode.http-address` to `0.0.0.0:50070` *on the NameNode machine only*, and set it to a real, routable address on the Secondary NameNode machine. The different addresses are needed in this case because HDFS uses `dfs.namenode.http-address` for two different purposes: it defines both the address the NameNode binds to, and the address the Secondary NameNode connects to for checkpointing. Using `0.0.0.0` on the NameNode allows the NameNode to bind to all its local addresses, while using the externally-routable address on the the Secondary NameNode provides the Secondary NameNode with a real address to connect to.

For more information, see Multi-host SecondaryNameNode Configuration.

More about the Secondary NameNode

- The NameNode stores the HDFS metadata information in RAM to speed up interactive lookups and modifications of the metadata.
- For reliability, this information is flushed to disk periodically. To ensure that these writes are not a speed bottleneck, only the list of modifications is written to disk, not a full snapshot of the current filesystem. The list of modifications is appended to a file called `edits`.
- Over time, the `edits` log file can grow quite large and consume large amounts of disk space.
- When the NameNode is restarted, it takes the HDFS system state from the `fsimage` file, then applies the contents of the `edits` log to construct an accurate system state that can be loaded into the NameNode's RAM. If you restart a large cluster that has run for a long period with no Secondary NameNode, the `edits` log may be quite large, and so it can take some time to reconstruct the system state to be loaded into RAM.

When the Secondary NameNode is configured, it periodically constructs a checkpoint by compacting the information in the edits log and merging it with the most recent `fsimage` file; it then clears the edits log. So, when the NameNode restarts, it can use the latest checkpoint and apply the contents of the smaller edits log. The interval between checkpoints is determined by the checkpoint period (`dfs.namenode.checkpoint.period`) or the number of edit transactions (`dfs.namenode.checkpoint.txns`). The default checkpoint period is one hour, and the default number of edit transactions before a checkpoint is 1,000,000. The SecondaryNameNode will checkpoint in an hour if there have not been 1,000,000 edit transactions within the hour; it will checkpoint after 1,000,000 transactions have been committed if they were committed in under one hour.

Secondary NameNode Parameters

The behavior of the Secondary NameNode is controlled by the following parameters in `hdfs-site.xml`.

- `dfs.namenode.checkpoint.check.period`
- `dfs.namenode.checkpoint.txns`
- `dfs.namenode.checkpoint.dir`
- `dfs.namenode.checkpoint.edits.dir`
- `dfs.namenode.num.checkpoints.retained`

See http://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml for details.

Enabling Trash

Important:

The trash feature is disabled by default. Cloudera recommends that you enable it on all production clusters.

The Hadoop trash feature helps prevent accidental deletion of files and directories. If trash is enabled and a file or directory is deleted using the Hadoop shell, the file is moved to the `.Trash` directory in the user's home directory instead of being deleted. Deleted files are initially moved to the `Current` sub-directory of the `.Trash` directory, and their original path is preserved. If trash checkpointing is enabled, the `Current` directory is periodically renamed using a timestamp. Files in `.Trash` are permanently removed after a user-configurable time delay. Files and directories in the trash can be restored simply by moving them to a location outside the `.Trash` directory.

Note:

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 is configured with the following properties in the `core-site.xml` file:

CDH Parameter	Value	Description
`fs.trash.interval`	*minutes or* 0	The number of minutes after which a trash checkpoint directory is deleted. This option can be configured both on the server and the client. ▪ If trash is enabled on the server configuration, then the value configured on the server is used and the client configuration is ignored. ▪ If trash is disabled in the server configuration, then the client side configuration is checked. ▪ If the value of this property is zero (the default), then the trash feature is disabled.
`fs.trash.checkpoint.interval`	*minutes or* 0	The number of minutes between trash checkpoints. Every time the checkpointer runs on the NameNode, it creates a new checkpoint of the "Current" directory and removes checkpoints older than `fs.trash.interval` minutes. This value should be smaller than or equal to `fs.trash.interval`. This option is configured on the server. If configured to zero (the default), then the value is set to the value of `fs.trash.interval`.

For example, to enable trash so that files deleted using the Hadoop shell are not deleted for 24 hours, set the value of the `fs.trash.interval` property in the server's `core-site.xml` file to a value of `1440`.

Note:

The period during which a file remains in the trash starts when the file is moved to the trash, not when the file is last modified.

Configuring Storage-Balancing for the 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.

To configure storage balancing, set the following properties in `hdfs-site.xml`.

Note: Keep in mind that if usage is markedly imbalanced among a given DataNode's storage volumes when you enable storage balancing, throughput on that DataNode will be affected initially, as writes are disproportionately directed to the under-utilized volumes.

Property	Value
`dfs.datanode.fsdataset. volume.choosing.policy`	`org.apache.hado`
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold`	10737418240 (defa
`dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction`	0.75 (default)

Enabling WebHDFS

Note:

To configure HttpFs instead, see HttpFS Installation on page 259.

If you want to use WebHDFS, you must first enable it.

To enable WebHDFS:

Set the following property in `hdfs-site.xml`:

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

To enable numeric usernames in WebHDFS:

By default, WebHDFS supports the following username pattern:

```
^[A-Za-z_][A-Za-z0-9._-]*[$]?$
```

You can override the default username pattern by setting the `dfs.webhdfs.user.provider.user.pattern` property in `hdfs-site.xml`. For example, to allow numerical usernames, the property can be set as follows:

```
<property>
  <name>dfs.webhdfs.user.provider.user.pattern</name>
  <value>^[A-Za-z0-9_][A-Za-z0-9._-]*[$]?$</value>
</property>
```

Important: The username pattern should be compliant with the requirements of the operating system in use. Hence, Cloudera recommends you use the default pattern and avoid modifying the `dfs.webhdfs.user.provider.user.pattern` property when possible.

Note:

- To use WebHDFS in a secure cluster, you must set additional properties to configure secure WebHDFS. For instructions, see the CDH 5 Security Guide.
- When you use WebHDFS in a high-availability (HA) configuration, you must supply the value of `dfs.nameservices` in the WebHDFS URI, rather than the address of a particular NameNode; for example:

  `hdfs dfs -ls webhdfs://nameservice1/`, *not*

  `hdfs dfs -ls webhdfs://server1.myent.myco.com:20101/`

Configuring LZO

If you have installed LZO, configure it as follows.

To configure LZO:

Set the following property in `core-site.xml`.

Note:

If you copy and paste the *value* string, make sure you remove the line-breaks and carriage returns, which are included below because of page-width constraints.

```
<property>
  <name>io.compression.codecs</name>

<value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec,
org.apache.hadoop.io.compress.BZip2Codec,com.hadoop.compression.lzo.LzoCodec,
com.hadoop.compression.lzo.LzopCodec,org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

For more information about LZO, see Using LZO Compression.

Start HDFS

To deploy HDFS now, proceed as follows.

1. Deploy the configuration.
2. Start HDFS.
3. Create the `/tmp` directory.

Deploy the configuration

To deploy your configuration to your entire cluster:

1. Push your custom directory (for example `/etc/hadoop/conf.my_cluster`) to each node in your cluster; for example:

```
$ scp -r /etc/hadoop/conf.my_cluster
myuser@myCDHnode-<n>.mycompany.com:/etc/hadoop/conf.my_cluster
```

2. Manually set `alternatives` on each node to point to that directory, as follows.

   To manually set the configuration on Red Hat-compatible systems:

```
$ sudo alternatives --verbose --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50
$ sudo alternatives --set hadoop-conf /etc/hadoop/conf.my_cluster
```

   To manually set the configuration on Ubuntu and SLES systems:

```
$ sudo update-alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.my_cluster 50
$ sudo update-alternatives --set hadoop-conf /etc/hadoop/conf.my_cluster
```

   For more information on alternatives, see the `update-alternatives(8)` man page on Ubuntu and SLES systems or the `alternatives(8)` man page On Red Hat-compatible systems.

Start HDFS

Start HDFS on each node in the cluster, as follows:

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

Note:

This starts all the CDH services installed on the node. This is normally what you want, but you can start services individually if you prefer.

Create the /tmp directory

Important:

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

Create the `/tmp` directory after HDFS is up and running, and set its permissions to 1777 (`drwxrwxrwt`), as follows:

```
$ sudo -u hdfs hadoop fs -mkdir /tmp
$ sudo -u hdfs hadoop fs -chmod -R 1777 /tmp
```

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>`

Deploy YARN or MRv1

To to deploy MRv1 or YARN, and start HDFS services if you have not already done so, see

- Deploying MapReduce v2 (YARN) on a Cluster on page 135 *or*
- Deploying MapReduce v1 (MRv1) on a Cluster on page 130

Deploying MapReduce v1 (MRv1) on a Cluster

This section describes configuration and startup tasks for MRv1 clusters only.

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade performance and may result in an unstable cluster deployment.

- Follow the instructions on this page to deploy MapReduce v1 (MRv1).
- If you have installed YARN and want to deploy it instead of MRv1, follow these instructions instead of the ones below.
- If you have installed CDH 5 from tarballs, the default deployment is YARN.

Important:

Do these tasks after you have configured and deployed HDFS:

1. Configure properties for MRv1 clusters
2. Configure local storage directories for use by MRv1 daemons
3. Configure a health check script for DataNode processes
4. Configure JobTracker Recovery
5. If necessary, deploy the configuration
6. If necessary, start HDFS
7. Create the HDFS `/tmp directory`
8. Create MapReduce `/var directories`
9. Verify the HDFS File Structure
10. Create and configure the `mapred.system.dir directory in HDFS`
11. Start MapReduce
12. Create a Home Directory for each MapReduce User
13. Configure the Hadoop daemons to start at boot time

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Step 1: Configuring Properties for MRv1 Clusters

Note:

Edit these files in the custom directory you created when you copied the Hadoop configuration.

For instructions on configuring a highly available JobTracker, see Configuring High Availability for the JobTracker (MRv1); you need to configure `mapred.job.tracker` differently in that case, and you must not use the port number.

Property	Configuration File	Description
`mapred.job.tracker`	`conf/mapred-site.xml`	If you plan to run your cluster with MRv1 daemons you need to specify the hostname and (optionally) port of the JobTracker's RPC server, in the form <host>:<port>. See Configuring Ports for CDH 5 for the default port. If the value is set to `local`, the default, the JobTracker runs on demand when you run a MapReduce job; do not try to start the JobTracker yourself in this case. Note: if you specify the host (rather than using `local`) this **must** be the hostname (for example mynamenode) not the IP address.

Sample configuration:

mapred-site.xml:

```
<property>
 <name>mapred.job.tracker</name>
 <value>jobtracker-host.company.com:8021</value>
</property>
```

Step 2: Configure Local Storage Directories for Use by MRv1 Daemons

For MRv1, you need to configure an additional property in the `mapred-site.xml` file.

Property	Configuration File Location	Description
`mapred.local.dir`	`mapred-site.xml` on each TaskTracker	This property specifies the directories where the TaskTracker will store temporary data and intermediate map output files while running MapReduce jobs. Cloudera recommends that this property specifies a directory on each of the JBOD mount points; for example, `/data/1/mapred/local` through `/data/N/mapred/local`.

Sample configuration:

mapred-site.xml on each TaskTracker:

```
<property>
 <name>mapred.local.dir</name>
 <value>/data/1/mapred/local,/data/2/mapred/local,/data/3/mapred/local</value>
</property>
```

After specifying these directories in the `mapred-site.xml` file, you must create the directories and assign the correct file permissions to them on each node in your cluster.

To configure local storage directories for use by MapReduce:

In the following instructions, local path examples are used to represent Hadoop parameters. The `mapred.local.dir` parameter is represented by the `/data/1/mapred/local`, `/data/2/mapred/local`, `/data/3/mapred/local`, and `/data/4/mapred/local` path examples. Change the path examples to match your configuration.

1. Create the `mapred.local.dir` local directories:

   ```
   $ sudo mkdir -p /data/1/mapred/local /data/2/mapred/local /data/3/mapred/local
   /data/4/mapred/local
   ```

2. Configure the owner of the `mapred.local.dir` directory to be the `mapred` user:

   ```
   $ sudo chown -R mapred:hadoop /data/1/mapred/local /data/2/mapred/local
   /data/3/mapred/local /data/4/mapred/local
   ```

   The correct owner and permissions of these local directories are:

Owner	Permissions
`mapred:hadoop`	drwxr-xr-x

Step 3: Configure a Health Check Script for DataNode Processes

In CDH releases before CDH 4, the failure of a single `mapred.local.dir` caused the MapReduce TaskTracker process to shut down, resulting in the machine not being available to execute tasks. In CDH 5, as in CDH 4, the TaskTracker process will continue to execute tasks as long as it has a single functioning `mapred.local.dir` available. No configuration change is necessary to enable this behavior.

Because a TaskTracker that has few functioning local directories will not perform well, Cloudera recommends configuring a health script that checks if the DataNode process is running (if configured as described under Configuring DataNodes to Tolerate Local Storage Directory Failure, the DataNode will shut down after the configured number of directory failures). Here is an example health script that exits if the DataNode process is not running:

```
#!/bin/bash
if ! jps | grep -q DataNode ; then
 echo ERROR: datanode not up
fi
```

In practice, the `dfs.data.dir` and `mapred.local.dir` are often configured on the same set of disks, so a disk failure will result in the failure of both a `dfs.data.dir` and `mapred.local.dir`.

See the section titled "Configuring the Node Health Check Script" in the Apache cluster setup documentation for further details.

Step 4: Configure JobTracker Recovery

JobTracker recovery means that jobs that are running when JobTracker fails (for example, because of a system crash or hardware failure) are re-run when the JobTracker is restarted. Any jobs that were running at the time of the failure will be re-run from the beginning automatically.

A recovered job will have the following properties:

- It will have the same job ID as when it was submitted.
- It will run under the same user as the original job.
- It will write to the same output directory as the original job, overwriting any previous output.
- It will show as RUNNING on the JobTracker web page after you restart the JobTracker.

Enabling JobTracker Recovery

By default JobTracker recovery is off, but you can enable it by setting the property `mapreduce.jobtracker.restart.recover` to `true` in `mapred-site.xml`.

Step 5: If Necessary, Deploy your Custom Configuration to your Entire Cluster

Deploy the configuration on page 129 if you have not already done so.

Step 6: If Necessary, Start HDFS on Every Node in the Cluster

Start HDFS on page 128 if you have not already done so .

Step 7: If Necessary, Create the HDFS /tmp Directory

Create the /tmp directory on page 129 if you have not already done so.

Important:

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

Step 8: Create MapReduce /var directories

```
sudo -u hdfs hadoop fs -mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chmod 1777 /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chown -R mapred /var/lib/hadoop-hdfs/cache/mapred
```

Step 9: Verify the HDFS File Structure

```
$ sudo -u hdfs hadoop fs -ls -R /
```

You should see:

```
drwxrwxrwt   - hdfs supergroup          0 2012-04-19 15:14 /tmp
drwxr-xr-x   - hdfs     supergroup          0 2012-04-19 15:16 /var
drwxr-xr-x   - hdfs     supergroup          0 2012-04-19 15:16 /var/lib
drwxr-xr-x   - hdfs     supergroup          0 2012-04-19 15:16 /var/lib/hadoop-hdfs
drwxr-xr-x  - hdfs    supergroup         0 2012-04-19 15:16 /var/lib/hadoop-hdfs/cache
drwxr-xr-x   - mapred   supergroup          0 2012-04-19 15:19
/var/lib/hadoop-hdfs/cache/mapred
drwxr-xr-x   - mapred   supergroup          0 2012-04-19 15:29
/var/lib/hadoop-hdfs/cache/mapred/mapred
drwxrwxrwt   - mapred   supergroup          0 2012-04-19 15:33
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging
```

Step 10: Create and Configure the mapred.system.dir Directory in HDFS

After you start HDFS and create `/tmp`, but before you start the JobTracker (see the next step), you must also create the HDFS directory specified by the `mapred.system.dir` parameter (by default `${hadoop.tmp.dir}/mapred/system` and configure it to be owned by the `mapred` user.

To create the directory in its default location:

```
$ sudo -u hdfs hadoop fs -mkdir /tmp/mapred/system
$ sudo -u hdfs hadoop fs -chown mapred:hadoop /tmp/mapred/system
```

Important:

If you create the `mapred.system.dir` directory in a different location, specify that path in the `conf/mapred-site.xml` file.

When starting up, MapReduce sets the permissions for the `mapred.system.dir` directory to drwx------, assuming the user `mapred` owns that directory.

Step 11: Start MapReduce

To start MapReduce, start the TaskTracker and JobTracker services

On each TaskTracker system:

```
$ sudo service hadoop-0.20-mapreduce-tasktracker start
```

On the JobTracker system:

```
$ sudo service hadoop-0.20-mapreduce-jobtracker start
```

Step 12: Create a Home Directory for each MapReduce User

Create a home directory for each MapReduce user. It is best to do this on the NameNode; for example:

```
$ sudo -u hdfs hadoop fs -mkdir  /user/<user>
$ sudo -u hdfs hadoop fs -chown <user> /user/<user>
```

where <user> is the Linux username of each user.

Alternatively, you can log in as each Linux user (or write a script to do so) and create the home directory as follows:

```
sudo -u hdfs hadoop fs -mkdir /user/$USER
sudo -u hdfs hadoop fs -chown $USER /user/$USER
```

Step 13: Set `HADOOP_MAPRED_HOME`

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

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

Configure the Hadoop Daemons to Start at Boot Time

See Configuring the Hadoop Daemons to Start at Boot Time.

Deploying MapReduce v2 (YARN) on a Cluster

This section describes configuration tasks for YARN clusters only, and is specifically tailored for administrators who have installed YARN from packages.

Important:

Do the following tasks after you have configured and deployed HDFS:

1. Configure properties for YARN clusters
2. Configure YARN daemons
3. Configure the History Server
4. Configure the Staging Directory
5. Deploy your custom configuration to your entire cluster
6. Start HDFS
7. Create the HDFS `/tmp directory`
8. Create the History Directory and Set Permissions
9. Create Log Directories
10. Verify the HDFS File Structure
11. Start YARN and the MapReduce JobHistory Server
12. Create a home directory for each MapReduce user
13. Configure the Hadoop daemons to start at boot time

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

About MapReduce v2 (YARN)

The default installation in CDH 5 is MapReduce 2.*x* (MRv2) built on the YARN framework. In this document we usually refer to this new version as **YARN**. The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks. For details of the new architecture, see Apache Hadoop NextGen MapReduce (YARN).

See also Selecting Appropriate JAR files for your MRv1 and YARN Jobs.

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not supported; it will degrade performance and may result in an unstable cluster deployment.

- If you have installed YARN from packages, follow the instructions below to deploy it. (To deploy MRv1 instead, see Deploying MapReduce v1 (MRv1) on a Cluster.)
- If you have installed CDH 5 from tarballs, the default deployment is YARN. Keep in mind that the instructions on this page are tailored for a deployment following installation from packages.

Step 1: Configure Properties for YARN Clusters

Note:

Edit these files in the custom directory you created when you copied the Hadoop configuration. When you have finished, you will push this configuration to all the nodes in the cluster; see Step 5.

Property	Configuration File	Description
`mapreduce.framework.name`	`mapred-site.xml`	If you plan on running YARN, you must set this property to the value of `yarn`.

Sample Configuration:

mapred-site.xml:

```
<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
</property>
```

Step 2: Configure YARN daemons

Configure the following services: ResourceManager (on a dedicated host) and NodeManager (on every host where you plan to run MapReduce v2 jobs).

The following table shows the most important properties that you must configure for your cluster in `yarn-site.xml`

Property	Recommended value	Description
`yarn.nodemanager.aux-services`	`mapreduce_shuffle`	Shuffle service that needs to be set for Map Reduce applications.
`yarn.resourcemanager.hostname`	`resourcemanager.company.com`	The following properties will be set to their default ports on this host: `yarn.resourcemanager.address`, `yarn.resourcemanager.admin.address`, `yarn.resourcemanager.scheduler.address`, `yarn.resourcemanager.resource-tracker.address`, `yarn.resourcemanager.webapp.address`

Property	Recommended value	Description
`yarn.application.classpath`	`$HADOOP_CONF_DIR, $HADOOP_COMMON_HOME/*, $HADOOP_COMMON_HOME/lib/*, $HADOOP_HDFS_HOME/*, $HADOOP_HDFS_HOME/lib/*, $HADOOP_MAPRED_HOME/*, $HADOOP_MAPRED_HOME/lib/*, $HADOOP_YARN_HOME/*, $HADOOP_YARN_HOME/lib/*`	Classpath for typical applications.
`yarn.log.aggregation-enable`	`true`	

Next, you need to specify, create, and assign the correct permissions to the local directories where you want the YARN daemons to store data.

You specify the directories by configuring the following two properties in the `yarn-site.xml` file on all cluster nodes:

Property	Description
`yarn.nodemanager.local-dirs`	Specifies the URIs of the directories where the NodeManager stores its localized files. All of the files required for running a particular YARN application will be put here for the duration of the application run. Cloudera recommends that this property specify a directory on each of the JBOD mount points; for example, `file:///data/1/yarn/local` through `/data/N/yarn/local`.
`yarn.nodemanager.log-dirs`	Specifies the URIs of the directories where the NodeManager stores container log files. Cloudera recommends that this property specify a directory on each of the JBOD mount points; for example, `file:///data/1/yarn/logs` through `file:///data/N/yarn/logs`.
`yarn.nodemanager.remote-app-log-dir`	Specifies the URI of the directory where logs are aggregated. Set the value to *either* `hdfs://`*`namenode-host.company.com`*`:8020/var/log/hadoop-yarn/apps`, using the fully-qualified domain name of your NameNode host, *or* `hdfs:/var/log/hadoop-yarn/apps`. See also Step 9.

Here is an example configuration:

yarn-site.xml:

```
<property>
<property>
  <name>yarn.resourcemanager.hostname</name>
  <value>resourcemanager.company.com</value>
</property>
<property>
  <description>Classpath for typical applications.</description>
  <name>yarn.application.classpath</name>
  <value>
      $HADOOP_CONF_DIR,
        $HADOOP_COMMON_HOME/*,$HADOOP_COMMON_HOME/lib/*,
        $HADOOP_HDFS_HOME/*,$HADOOP_HDFS_HOME/lib/*,
        $HADOOP_MAPRED_HOME/*,$HADOOP_MAPRED_HOME/lib/*,
        $HADOOP_YARN_HOME/*,$HADOOP_YARN_HOME/lib/*
    </value>
  </property>
  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
  </property>
  <property>
    <name>yarn.nodemanager.local-dirs</name>

<value>file:///data/1/yarn/local,file:///data/2/yarn/local,file:///data/3/yarn/local</value>

  </property>
  <property>
    <name>yarn.nodemanager.log-dirs</name>

<value>file:///data/1/yarn/logs,file:///data/2/yarn/logs,file:///data/3/yarn/logs</value>

  </property>
  <property>
  </property>
    <name>yarn.log.aggregation-enable</name>
    <value>true</value>
  <property>
    <description>Where to aggregate logs</description>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>hdfs://<namenode-host.company.com>:8020/var/log/hadoop-yarn/apps</value>
  </property>
```

After specifying these directories in the `yarn-site.xml` file, you must create the directories and assign the correct file permissions to them on each node in your cluster.

In the following instructions, local path examples are used to represent Hadoop parameters. Change the path examples to match your configuration.

To configure local storage directories for use by YARN:

1. Create the `yarn.nodemanager.local-dirs` local directories:

   ```
   $ sudo mkdir -p /data/1/yarn/local /data/2/yarn/local /data/3/yarn/local
   /data/4/yarn/local
   ```

2. Create the `yarn.nodemanager.log-dirs` local directories:

   ```
   $ sudo mkdir -p /data/1/yarn/logs /data/2/yarn/logs /data/3/yarn/logs
   /data/4/yarn/logs
   ```

3. Configure the owner of the `yarn.nodemanager.local-dirs` directory to be the yarn user:

   ```
   $ sudo chown -R yarn:yarn /data/1/yarn/local /data/2/yarn/local /data/3/yarn/local
    /data/4/yarn/local
   ```

4. Configure the owner of the `yarn.nodemanager.log-dirs` directory to be the yarn user:

   ```
   $ sudo chown -R yarn:yarn /data/1/yarn/logs /data/2/yarn/logs /data/3/yarn/logs
   /data/4/yarn/logs
   ```

Here is a summary of the correct owner and permissions of the local directories:

Directory	Owner	Permissions
`yarn.nodemanager.local-dirs`	`yarn:yarn`	drwxr-xr-x

Directory	Owner	Permissions
`yarn.nodemanager.log-dirs`	`yarn:yarn`	drwxr-xr-x

Step 3: Configure the History Server

If you have decided to run YARN on your cluster instead of MRv1, you should also run the MapReduce JobHistory Server. The following table shows the most important properties that you must configure in `mapred-site.xml`.

Property	Recommended value	Description
`mapreduce.jobhistory.address`	`historyserver.company.com:10020`	The address of the JobHistory Server `host:port`
`mapreduce.jobhistory.webapp.address`	`historyserver.company.com:19888`	The address of the JobHistory Server web application `host:port`

In addition, make sure proxying is enabled for the `mapred` user; configure the following properties in `core-site.xml`:

Property	Recommended value	Description
`hadoop.proxyuser.mapred.groups`	*	Allows the `mapred`user to move files belonging to users in these groups
`hadoop.proxyuser.mapred.hosts`	*	Allows the `mapred`user to move files belonging on these hosts

Step 4: Configure the Staging Directory

YARN requires a staging directory for temporary files created by running jobs. By default it creates `/tmp/hadoop-yarn/staging` with restrictive permissions that may prevent your users from running jobs. To forestall this, you should configure and create the staging directory yourself; in the example that follows we use `/user`:

1. Configure `yarn.app.mapreduce.am.staging-dir` in `mapred-site.xml`:

```
<property>
    <name>yarn.app.mapreduce.am.staging-dir</name>
    <value>/user</value>
</property>
```

2. Once HDFS is up and running, you will create this directory and a `history` subdirectory under it (see Step 8).

Alternatively, you can do the following:

1. Configure `mapreduce.jobhistory.intermediate-done-dir` and `mapreduce.jobhistory.done-dir` in `mapred-site.xml`.
2. Create these two directories.
3. Set permissions on `mapreduce.jobhistory.intermediate-done-dir` to 1777.
4. Set permissions on `mapreduce.jobhistory.done-dir` to 750.

If you configure `mapreduce.jobhistory.intermediate-done-dir` and `mapreduce.jobhistory.done-dir` as above, you can skip Step 8.

Step 5: If Necessary, Deploy your Custom Configuration to your Entire Cluster

Deploy the configuration on page 129 if you have not already done so.

Step 6: If Necessary, Start HDFS on Every Node in the Cluster

Start HDFS on page 128 if you have not already done so.

Step 7: If Necessary, Create the HDFS `/tmp` Directory

Create the /tmp directory on page 129 if you have not already done so.

Important:

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

Step 8: Create the history Directory and Set Permissions and Owner

This is a subdirectory of the staging directory you configured in Step 4. In this example we're using `/user/history`. Create it and set permissions as follows:

```
sudo -u hdfs hadoop fs -mkdir -p /user/history
sudo -u hdfs hadoop fs -chmod -R 1777 /user/history
sudo -u hdfs hadoop fs -chown mapred:hadoop /user/history
```

Step 9: Create Log Directories

Note:

See also Step 2.

Create the `/var/log/hadoop-yarn` directory and set ownership:

```

sudo -u hdfs hadoop fs -mkdir -p /var/log/hadoop-yarn
sudo -u hdfs hadoop fs -chown yarn:mapred /var/log/hadoop-yarn
```

Note:

You need to create this directory because it is the parent of `/var/log/hadoop-yarn/apps` which is explicitly configured in `yarn-site.xml`.

Step 10: Verify the HDFS File Structure:

```
$ sudo -u hdfs hadoop fs -ls -R /
```

You should see:

```
drwxrwxrwt   - hdfs supergroup          0 2012-04-19 14:31 /tmp
drwxr-xr-x   - hdfs supergroup          0 2012-05-31 10:26 /user
drwxrwxrwt   - yarn supergroup          0 2012-04-19 14:31 /user/history
drwxr-xr-x   - hdfs   supergroup        0 2012-05-31 15:31 /var
drwxr-xr-x   - hdfs   supergroup        0 2012-05-31 15:31 /var/log
drwxr-xr-x   - yarn   mapred            0 2012-05-31 15:31 /var/log/hadoop-yarn
```

Step 11: Start YARN and the MapReduce JobHistory Server

To start YARN, start the ResourceManager and NodeManager services:

Note:

Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

```
$ sudo service hadoop-yarn-resourcemanager start
```

On each NodeManager system (typically the same ones where DataNode service runs):

```
$ sudo service hadoop-yarn-nodemanager start
```

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

```
$ sudo service hadoop-mapreduce-historyserver start
```

Step 12: Create a Home Directory for each MapReduce User

Create a home directory for each MapReduce user. It is best to do this on the NameNode; for example:

```
$ sudo -u hdfs hadoop fs -mkdir  /user/<user>
$ sudo -u hdfs hadoop fs -chown <user> /user/<user>
```

where <user> is the Linux username of each user.

Alternatively, you can log in as each Linux user (or write a script to do so) and create the home directory as follows:

```
sudo -u hdfs hadoop fs -mkdir /user/$USER
sudo -u hdfs hadoop fs -chown $USER /user/$USER
```

Step 13: Configure the Hadoop Daemons to Start at Boot Time

See Configuring the Hadoop Daemons to Start at Boot Time.

Configuring the Daemons to Start on Boot

Important:

Make sure you are not trying to run MRv1 and YARN on the same set of nodes at the same time. This is not recommended; it will degrade your performance and may result in an unstable MapReduce cluster deployment.

To start the Hadoop daemons at boot time and on restarts, enable their `init` scripts on the systems on which the services will run, using the `chkconfig` tool. See Configuring init to Start Core Hadoop System Services.

Non-core services can also be started at boot time; after you install the non-core components, see Configuring init to Start Non-core Hadoop System Services for instructions.

Tips and Guidelines

Selecting Appropriate JAR files for your MRv1 and YARN Jobs

Each implementation of the CDH 5 MapReduce framework (MRv1 and YARN) consists of the artifacts (JAR files) that provide MapReduce functionality as well as auxiliary utility artifacts that are used during the course of the MapReduce job. When you submit a job either explicitly (using the Hadoop launcher script) or implicitly (via Java implementations) it is extremely important that you make sure that you reference utility artifacts that come with the same version of MapReduce implementation that is running on your cluster. The following table summarizes the names and location of these artifacts:

Name	MRv1 location	YARN location
streaming	`/usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh<version>.jar`	`/usr/lib/hadoop-mapreduce/hadoop-streaming.jar`
rumen	N/A	`/usr/lib/hadoop-mapreduce/hadoop-rumen.jar`
hadoop examples	`/usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar`	`/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar`
distcp v1	`/usr/lib/hadoop-0.20-mapreduce/hadoop-tools.jar`	`/usr/lib/hadoop-mapreduce/hadoop-extras.jar`
distcp v2	N/A	`/usr/lib/hadoop-mapreduce/hadoop-distcp.jar`
hadoop archives	`/usr/lib/hadoop-0.20-mapreduce/hadoop-tools.jar`	`/usr/lib/hadoop-mapreduce/hadoop-archives.jar`

Improving Performance

This section provides solutions to some performance problems, and describes configuration best practices.

Important:

If you are running CDH over 10Gbps Ethernet, improperly set network configuration or improperly applied NIC firmware or drivers can noticeably degrade performance. Work with your network engineers and hardware vendors to make sure that you have the proper NIC firmware, drivers, and configurations in place and that your network performs properly. Cloudera recognizes that network setup and upgrade are challenging problems, and will make best efforts to share any helpful experiences.

Disabling Transparent Hugepage Compaction

Most Linux platforms supported by CDH 5 include a feature called ***transparent hugepage compaction*** which interacts poorly with Hadoop workloads and can seriously degrade performance.

Symptom: top and other system monitoring tools show a large percentage of the CPU usage classified as "system CPU". If system CPU usage is 30% or more of the total CPU usage, your system may be experiencing this issue.

What to do:

Note: In the following instructions, *defrag_file_pathname* depends on your operating system:

- Red Hat/CentOS: `/sys/kernel/mm/redhat_transparent_hugepage/defrag`
- Ubuntu/Debian, OEL, SLES: `/sys/kernel/mm/transparent_hugepage/defrag`

1. To see whether transparent hugepage compaction is enabled, run the following command and check the output:

   ```
   $ cat defrag_file_pathname
   ```

   - `[always] never` means that transparent hugepage compaction is enabled.
   - `always [never]` means that transparent hugepage compaction is disabled.

2. To disable transparent hugepage compaction, add the following command to `/etc/rc.local`:

   ```
    echo never > defrag_file_pathname
   ```

You can also disable transparent hugepage compaction interactively (but remember this will not survive a reboot).

To disable transparent hugepage compaction temporarily as root:

```
# echo 'never' > defrag_file_pathname
```

To disable transparent hugepage compaction temporarily using sudo:

```
$ sudo sh -c "echo 'never' > defrag_file_pathname"
```

Setting the vm.swappiness Linux Kernel Parameter

`vm.swappiness` is a Linux Kernel Parameter that controls how aggressively memory pages are swapped to disk. It can be set to a value between 0-100; the higher the value, the more aggressive the kernel is in seeking out inactive memory pages and swapping them to disk.

You can see what value `vm.swappiness` is currently set to by looking at `/proc/sys/vm`; for example:

```
cat /proc/sys/vm/swappiness
```

On most systems, it is set to 60 by default. This is not suitable for Hadoop clusters nodes, because it can cause processes to get swapped out even when there is free memory available. This can affect stability and performance, and may cause problems such as lengthy garbage collection pauses for important system daemons. Cloudera recommends that you set this parameter to 0; for example:

```
# sysctl -w vm.swappiness=0
```

Improving Performance in Shuffle Handler and IFile Reader

The MapReduce shuffle handler and IFile reader use native Linux calls (`posix_fadvise`(2) and `sync_data_range`) on Linux systems with Hadoop native libraries installed. The subsections that follow provide details.

Shuffle Handler

You can improve MapReduce Shuffle Handler Performance by enabling shuffle readahead. This causes the TaskTracker or Node Manager to pre-fetch map output before sending it over the socket to the reducer.

- To enable this feature for YARN, set the `mapreduce.shuffle.manage.os.cache` property to `true` (default). To further tune performance, adjust the value of the `mapreduce.shuffle.readahead.bytes` property. The default value is 4MB.
- To enable this feature for MRv1, set the `mapred.tasktracker.shuffle.fadvise` property to `true` (default). To further tune performance, adjust the value of the `mapred.tasktracker.shuffle.readahead.bytes` property. The default value is 4MB.

IFile Reader

Enabling IFile readahead increases the performance of merge operations. To enable this feature for either MRv1 or YARN, set the `mapreduce.ifile.readahead` property to `true` (default). To further tune the performance, adjust the value of the `mapreduce.ifile.readahead.bytes` property. The default value is 4MB.

Best Practices for MapReduce Configuration

The configuration settings described below can reduce inherent latencies in MapReduce execution. You set these values in `mapred-site.xml`.

Send a heartbeat as soon as a task finishes

Set the `mapreduce.tasktracker.outofband.heartbeat` property to `true` to let the TaskTracker send an out-of-band heartbeat on task completion to reduce latency; the default value is `false`:

```
<property>
    <name>mapreduce.tasktracker.outofband.heartbeat</name>
    <value>true</value>
</property>
```

Reduce the interval for JobClient status reports on single node systems

The `jobclient.progress.monitor.poll.interval` property defines the interval (in milliseconds) at which JobClient reports status to the console and checks for job completion. The default value is 1000 milliseconds; you may want to set this to a lower value to make tests run faster on a single-node cluster. Adjusting this value on a large production cluster may lead to unwanted client-server traffic.

```
<property>
    <name>jobclient.progress.monitor.poll.interval</name>
    <value>10</value>
</property>
```

Tune the JobTracker heartbeat interval

Tuning the minimum interval for the TaskTracker-to-JobTracker heartbeat to a smaller value may improve MapReduce performance on small clusters.

```
<property>
    <name>mapreduce.jobtracker.heartbeat.interval.min</name>
    <value>10</value>
</property>
```

Start MapReduce JVMs immediately

The `mapred.reduce.slowstart.completed.maps` property specifies the proportion of Map tasks in a job that must be completed before any Reduce tasks are scheduled. For small jobs that require fast turnaround, setting this value to 0 can improve performance; larger values (as high as 50%) may be appropriate for larger jobs.

```
<property>
    <name>mapred.reduce.slowstart.completed.maps</name>
    <value>0</value>
</property>
```

Best practices for HDFS Configuration

This section indicates changes you may want to make in `hdfs-site.xml`.

Improve Performance for Local Reads

Note:

Also known as **short-circuit local reads**, this capability is particularly useful for HBase and Cloudera Impala™. It improves the performance of node-local reads by providing a fast path that is enabled in this case. It 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 in order to use short-circuit local reads.

Configure the following properties in `hdfs-site.xml` as shown:

```
<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`.

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()` method.

To add JARs to the classpath, use `-libjars <jar1>,<jar2>`, which will copy the local JAR files to HDFS and then use the distributed cache mechanism to make sure they are available on the task nodes and are added to the task classpath.

The advantage of this over `JobConf.setJar` is that if the JAR is on a task node it won't need to be copied again if a second task from the same job runs on that node, though it will still need to be copied from the launch machine to HDFS.

Note:

`-libjars` works only if your MapReduce driver uses ToolRunner. If it doesn't, you would need to use the DistributedCache APIs (Cloudera does not recommend this).

For more information, see item 1 in the blog post How to Include Third-Party Libraries in Your MapReduce Job.

Changing the Logging Level on a Job (MRv1)

You can change the logging level for an individual job. You do this by setting the following properties in the job configuration (`JobConf`):

- mapreduce.map.log.level
- mapreduce.reduce.log.level

Valid values are `NONE`, `INFO`, `WARN`, `DEBUG`, `TRACE`, and `ALL`.

Example:

```
JobConf conf = new JobConf();
...

conf.set("mapreduce.map.log.level", "DEBUG");
conf.set("mapreduce.reduce.log.level", "TRACE");
...
```

Setting Quotas

As the system administrator, you can set quotas in HDFS for:

- The number of file and directory names used; *and*
- 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.
- 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.

Commands

To set space quotas on a directory:

```
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:

```
dfsadmin -clrSpaceQuota directory
```

You can specify multiple directories in a single command.

To set name quotas on a directory:

```
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:

```
dfsadmin -clrQuota directory
```

You can specify multiple directories in a single command.

For More Information

For more information, see the HDFS Quotas Guide.

Centralized Cache Management in HDFS

Overview

Centralized cache management in HDFS is an explicit caching mechanism that allows users to specify paths to be cached by HDFS. The NameNode will communicate with DataNodes that have the desired blocks on disk, and instruct them to cache the blocks in off-heap caches.

This has several advantages:

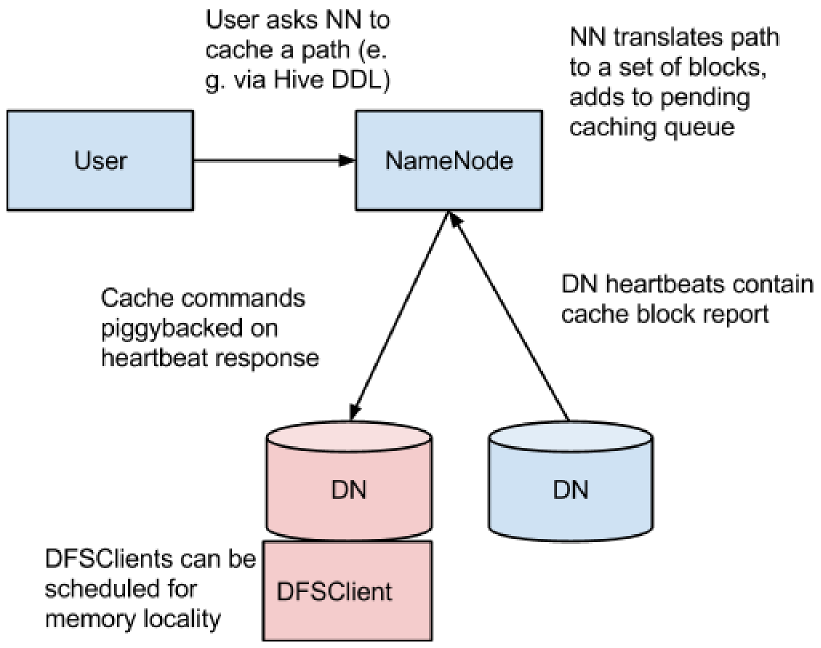
- Explicit pinning prevents frequently used data from being evicted from memory. This is particularly important when the size of the working set exceeds the size of main memory, which is common for many HDFS workloads.
- Since DataNode caches are managed by the NameNode, applications can query the set of cached block locations when making task placement decisions. Co-locating a task with a cached block replica improves read performance.
- When block has been cached by a DataNode, clients can use a new, more-efficient, zero-copy read API. Since checksum verification of cached data is done once by the DataNode, clients can incur essentially zero overhead when using this new API.
- Centralized caching can improve overall cluster memory utilization. When relying on the OS buffer cache at each DataNode, repeated reads of a block will result in all n replicas of the block being pulled into buffer cache. With centralized cache management, a user can explicitly pin only `m` of the n replicas, saving `n-m` memory.

Use Cases

Centralized cache management is most useful for files which are accessed repeatedly. For example, a fact table in Hive which is often used in joins is a good candidate for caching. On the other hand, caching the input of a one-year reporting query is probably less useful, since the historical data might be read only once.

Centralized cache management is also useful for mixed workloads with performance SLAs. Caching the working set of a high-priority workload insures that it does not contend for disk I/O with a low-priority workload.

Architecture


User asks NN to cache a path (e. g. via Hive DDL)
NN translates path to a set of blocks, adds to pending caching queue
User
NameNode
Cache commands piggybacked on heartbeat response
DN heartbeats contain cache block report
DN
DN
DFSClients can be scheduled for memory locality
DFSClient

In this architecture, the NameNode is responsible for coordinating all the DataNode off-heap caches in the cluster. The NameNode periodically receives a "cache report" from each DataNode which describes all the blocks cached on a given DataNode. The NameNode manages DataNode caches by piggybacking cache and uncache commands on the DataNode heartbeat.

The NameNode queries its set of cache directives to determine which paths should be cached. Cache directives are persistently stored in the fsimage and edit log, and can be added, removed, and modified using Java and command-line APIs. The NameNode also stores a set of cache pools, which are administrative entities used to group cache directives together for resource management and enforcing permissions.

The NameNode periodically rescans the namespace and active cache directories to determine which blocks need to be cached or uncached and assign caching to DataNodes. Rescans can also be triggered by user actions like adding or removing a cache directive or removing a cache pool.

We do not currently cache blocks which are under construction, corrupt, or otherwise incomplete. If a cache directive covers a symlink, the symlink target is not cached. Caching is currently done on a per-file basis, although we would like to add block-level granularity in the future.

Concepts

Cache Directives

A cache directive defines a path that should be cached. Paths can be either directories or files. Directories are cached non-recursively, meaning only files in the first-level listing of the directory will be cached. Directives also specify additional parameters, such as the cache replication factor and expiration time. The replication factor specifies the number of block replicas to cache. If multiple cache directives refer to the same file, the maximum cache replication factor is applied.

The expiration time is specified on the command line as a `time-to-live` (TTL), a relative expiration time in the future. After a cache directive expires, it is no longer considered by the NameNode when making caching decisions.

Cache Pool

A cache pool is an administrative entity used to manage groups of cache directives. Cache pools have UNIX-like permissions that restrict which users and groups have access to the pool. Write permissions allow users to add and remove cache directives to the pool. Read permissions allow users to list the cache directives in a pool, as well as additional metadata. Execute permissions are unused.

Cache pools are also used for resource management. Pools can enforce a maximum `limit`, which restricts the number of bytes that can be cached in aggregate by directives in the pool. Normally, the sum of the pool limits will approximately equal the amount of aggregate memory reserved for HDFS caching on the cluster. Cache pools also track a number of statistics to help cluster users determine what is and should be cached.

Pools also enforce a maximum time-to-live. This restricts the maximum expiration time of directives being added to the pool.

cacheadmin Command-Line Interface

On the command-line, administrators and users can interact with cache pools and directives via the `hdfs cacheadmin` subcommand. Cache directives are identified by a unique, non-repeating 64-bit integer ID. IDs will not be reused even if a cache directive is later removed. Cache pools are identified by a unique string name.

Cache Directive Commands

addDirective

Description: Add a new cache directive.

Usage: `hdfs cacheadmin -addDirective -path <path> -pool <pool-name> [-force] [-replication <replication>] [-ttl <time-to-live>]`

Where, `path`: A path to cache. The path can be a directory or a file.

`pool-name`: The pool to which the directive will be added. You must have write permission on the cache pool in order to add new directives.

`force`: Skips checking of cache pool resource limits.

`replication`: The cache replication factor to use. Defaults to 1.

`time-to-live`: Time period for which the directive is valid. Can be specified in seconds, minutes, hours, and days, e.g. 30m, 4h, 2d. The value `never` indicates a directive that never expires. If unspecified, the directive never expires.

removeDirective

Description: Remove a cache directive.

Usage: `hdfs cacheadmin -removeDirective <id>`

Where, `id`: The id of the cache directive to remove. You must have write permission on the pool of the directive in order to remove it. To see a list of PathBasedCache directive IDs, use the -listDirectives command.

removeDirectives

Description: Remove every cache directive with the specified path.

Usage: `hdfs cacheadmin -removeDirectives <path>`

Where, `path`: The path of the cache directives to remove. You must have write permission on the pool of the directive in order to remove it.

listDirectives

Description: List PathBasedCache directives.

Usage: `hdfs cacheadmin -listDirectives [-stats] [-path <path>] [-pool <pool>]`

Where, `path`: List only PathBasedCache directives with this path. Note that if there is a PathBasedCache directive for `path` in a cache pool that we don't 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, e.g. 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, e.g. 120s, 30m, 4h, 2d. By default, no maximum is set. A value of `never` specifies that there is no limit.

modifyPool

Description: Modifies 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, e.g. name, owner, group, permissions, etc.

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

In order 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:

- `dfs.datanode.max.locked.memory`: This determines the maximum amount of memory a DataNode will use for caching (in bytes). The "locked-in-memory size" ulimit (`ulimit -l`) of the DataNode user also needs to be increased to match this parameter (see OS Limits). When setting this value, remember that you will need space in memory for other things as well, such as the DataNode and application JVM heaps and the operating system page cache.

Optional

The following properties are not required, but may be specified for tuning:

- `dfs.namenode.path.based.cache.refresh.interval.ms`: The NameNode will use this as the amount of milliseconds between subsequent path cache rescans. This calculates the blocks to cache and each DataNode containing a replica of the block that should cache it. By default, this parameter is set to 300000, which is five minutes.
- `dfs.datanode.fsdatasetcache.max.threads.per.volume`: The DataNode will use 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 will use this as the amount of milliseconds between sending a full report of its cache state to the NameNode. By default, this parameter is set to 10000, which is 10 seconds.
- `dfs.namenode.path.based.cache.block.map.allocation.percent`: The percentage of the Java heap which we will allocate to the cached blocks map. The cached blocks map is a hash map which uses chained hashing. Smaller maps may be accessed more slowly if the number of cached blocks is large; larger maps will consume more memory. By default, this parameter is set to 0.25 percent.

OS Limits

If you get the error "Cannot start datanode because the configured max locked memory size... is more than the datanode's available RLIMIT_MEMLOCK ulimit," that means that the operating system is imposing a lower limit on the amount of memory that you can lock than what you have configured. To fix this, you must adjust the `ulimit -l` value that the DataNode runs with. Usually, this value is configured in `/etc/security/limits.conf`. However, it will vary depending on what operating system and distribution you are using.

You will know that you have correctly configured this value when you can run `ulimit -l` from the shell and get back either a higher value than what you have configured with `dfs.datanode.max.locked.memory`, or the string `unlimited`, indicating that there is no limit. Note that it's typical for `ulimit -l` to output the memory lock limit in KB, but `dfs.datanode.max.locked.memory` must be specified in bytes.

About HDFS Snapshots

CDH 5 provides an HDFS snapshot capability. A snapshot is a copy of all or part of the file system at a given point in time. Important characteristics of HDFS snapshots include:

- You can take a snapshot of the entire file system, or of any directory once you have made the directory ***snapshottable*** (`hdfs dfsadmin -allowSnapshot <path>`).
- There is no limit on the number of snapshottable directories.
- You can keep up to 65,536 snapshots of each directory.
- After you have taken the first snapshot, subsequent snapshots consume only as much disk space as the delta between data already in snapshot and the current state of the live file system.
- Snapshots are instantaneous and do not interfere with other HDFS operations.

For more information about snapshots, and instructions for taking and maintaining them, see HDFS Snapshots.

Crunch Installation

The Apache Crunch™ project develops and supports Java APIs that simplify the process of creating data pipelines on top of Apache Hadoop. The Crunch APIs are modeled after FlumeJava (PDF), which is the library that Google uses for building data pipelines on top of their own implementation of MapReduce.

The Apache Crunch Java library provides a framework for writing, testing, and running MapReduce pipelines. Its goal is to make pipelines that are composed of many user-defined functions simple to write, easy to test, and efficient to run. Running on top of Hadoop MapReduce and Apache Spark, the Apache Crunch library is a simple Java API for tasks like joining and data aggregation that are tedious to implement on plain MapReduce. The APIs are especially useful when processing data that does not fit naturally into relational model, such as time series, serialized object formats like protocol buffers or Avro records, and HBase rows and columns. For Scala users, there is the Scrunch API, which is built on top of the Java APIs and includes a REPL (read-eval-print loop) for creating MapReduce pipelines.

The following sections describe how to install Crunch:

- Crunch Prerequisites on page 155
- Crunch Packaging on page 155
- Installing and Upgrading Crunch on page 155
- Crunch Documentation on page 156

Crunch Prerequisites

- An operating system supported by CDH 5
- Oracle JDK

Crunch Packaging

The packaging options for installing Crunch are:

- RPM packages
- Debian packages

There are two Crunch packages:

- `crunch`: provides all the functionality of crunch allowing users to create data pipelines over execution engines like MapReduce, Spark, etc.
- `crunch-doc`: the documentation package.

Note: Crunch is also available as a parcel, included with the CDH 5 parcel. If you install CDH 5 with Cloudera Manager, Crunch will be installed automatically.

Installing and Upgrading Crunch

To install the Crunch packages:

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper/YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation on page 27 and Upgrading to CDH 5 on page 58.

To install or upgrade Crunch on a Red Hat system:

```
$ sudo yum install crunch
```

To install or upgrade Crunch on a SLES system:

```
$ sudo zypper install crunch
```

To install or upgrade Crunch on an Ubuntu or Debian system:

```
$ sudo apt-get install crunch
```

To use the Crunch documentation:

The Crunch docs are bundled in a `crunch-doc` package that should be installed separately.

```
$ sudo apt-get install crunch-doc
```

The contents of this package are saved under `/usr/share/doc/crunch*`.

After a package installation, the Crunch jars can be found in `/usr/lib/crunch`.

If you installed CDH 5 through Cloudera Manager, the CDH 5 parcel includes Crunch and the jars are installed automatically as part of the CDH 5 installation. By default the jars will be found in `/opt/cloudera/parcels/CDH/lib/crunch`.

Crunch Documentation

For more information about Crunch, see the following documentation:

- Getting Started with Crunch
- Apache Crunch User Guide

Flume Installation

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.

The following sections provide more information and instructions:

- Upgrading Flume on page 157
- Packaging
- Installing a Tarball
- Installing Packages
- Configuring Flume
- Verifying the Installation
- Running Flume
- Files Installed by the Packages
- Supported Sources, Sinks, and Channels
- Using and On-disk Encrypted File Channel
- Apache Flume Documentation

Upgrading Flume

Use the instructions that follow to upgrade Flume.

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Upgrading Flume from CDH 4 to CDH 5

Use one of the following sets of instructions to upgrade Flume from CDH 4 to CDH 5, depending on which version of Flume you have been running under CDH 4:

- If you are running Flume 1.x, use these instructions.
- If you are running Flume 0.9.x, use these instructions.

Upgrading Flume 1.x to CDH 5

Use the instructions that follow to upgrade Flume 1.x from CDH 4 to CDH 5. You must remove the CDH 4 version and then install the CDH 5 version.

Step 1: Remove Flume 1.x from your cluster.

1. Stop the Flume Node processes on each node where they are running:

```
$ sudo service flume-ng-agent stop
```

2. Uninstall the old Flume components:

On Red Hat-compatible systems:

```
$ sudo yum remove flume-ng agent flume-ng
```

On SLES systems:

```
$ sudo zypper remove flume-ng agent flume-ng
```

On Ubuntu systems:

```
$ sudo apt-get remove flume-ng agent flume-ng
```

Step 2. Install the new version of Flume

Follow the instructions in the rest of this document to install Flume 1.x under CDH 5.

Migrating from Flume 0.9.x under CDH 4 to Flume 1.x under CDH 5

Flume 1.x is a significant departure from Flume 0.9.x in its implementation although many of the original concepts are the same. If you're already familiar with Flume, here are some significant points.

- You still have sources and sinks and they still do the same thing. They are now connected by channels.
- Channels are pluggable, and dictate durability. Flume 1.x ships with an in-memory channel for fast, but non-durable event delivery. There are also JDBC-based and file-based channels for durable event delivery.
- There are no more logical or physical nodes. All physical nodes are "agents," and agents can run zero or more sources and sinks.
- There is no longer a Master, and no ZooKeeper dependency. At this time, Flume runs with a simple file-based configuration system.
- Just about everything is a plugin — some end user facing, some for tool and system developers.
- Thrift and Avro legacy Flume sources are provided to enable sending events from Flume 0.9.4 to Flume 1.x.

You must uninstall Flume 0.9.x and then install Flume 1.x, as follows.

Step 1: Remove Flume 0.9.x from your cluster.

1. Stop the Flume Node processes on each node where they are running:

   ```
   $ sudo service flume-node stop
   ```

2. Stop the Flume Master:

   ```
   $ sudo service flume-master stop
   ```

3. Uninstall the old Flume components:

   On Red Hat-compatible systems:

   ```
   $ sudo yum remove flume
   ```

   On SLES systems:

   ```
   $ sudo zypper remove flume
   ```

   On Ubuntu systems:

   ```
   $ sudo apt-get remove flume
   ```

Step 2. Install the new version of Flume

Follow the instructions in the rest of this document to install Flume 1.x from a tarball or packages.

Upgrading Flume from an Earlier CDH 5 release

These instructions assume that you are upgrading Sqoop as part of an upgrade to the latest CDH 5 release, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

To upgrade Sqoop from an earlier CDH 5 release, install the new version of Flume using one of the methods described below: Installing the Flume RPM or Debian Packages on page 160 or Installing the Flume Tarball on page 159 .

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Flume Packaging

There are currently three packaging options available for installing Flume:

- Tarball (.tar.gz)
- RPM packages
- Debian packages

Installing the Flume Tarball

The Flume tarball is a self-contained package containing everything needed to use Flume on a Unix-like system. To install Flume from the tarball, you unpack it in the appropriate directory.

Note:

The tarball does not come with any scripts suitable for running Flume as a service or daemon. This makes the tarball distribution appropriate for *ad hoc* installations and preliminary testing, but a more complete installation is provided by the binary RPM and Debian packages.

To install the Flume tarball on Linux-based systems:

1. Run the following commands, replacing the `(component_version)` with the current version numbers for Flume and CDH.

```
$ cd /usr/local/lib
$ sudo tar -zxvf <path_to_flume-ng-(Flume_version)-cdh(CDH_version).tar.gz>
$ sudo mv flume-ng-(Flume_version)-cdh(CDH_version) flume-ng
```

For example,

```
$ cd /usr/local/lib
$ sudo tar -zxvf <path_to_flume-ng-1.4.0-cdh5.0.0.tar.gz>
$ sudo mv flume-ng-1.4.0-cdh5.0.0 flume-ng
```

2. To complete the configuration of a tarball installation, you must set your PATH variable to include the bin/ subdirectory of the directory where you installed Flume. For example:

```
$ export PATH=/usr/local/lib/flume-ng/bin:$PATH
```

Installing the Flume RPM or Debian Packages

Installing the Flume RPM and Debian packages is more convenient than installing the Flume tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations
- Handle daemon startup and shutdown.

The Flume RPM and Debian packages consist of three packages:

- `flume-ng` — Everything you need to run Flume
- `flume-ng-agent` — Handles starting and stopping the Flume agent as a service
- `flume-ng-doc` — Flume documentation

All Flume installations require the common code provided by `flume-ng`.

- **Important:**

  If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install Flume. For instructions, see CDH 5 Installation on page 27.

To install Flume on Ubuntu and other Debian systems:

```
$ sudo apt-get install flume-ng
```

To install Flume On Red Hat-compatible systems:

```
$ sudo yum install flume-ng
```

To install Flume on SLES systems:

```
$ sudo zypper install flume-ng
```

You may also want to enable automatic start-up on boot. To do this, install the Flume agent.

To install the Flume agent so Flume starts automatically on boot on Ubuntu and other Debian systems:

```
$ sudo apt-get install flume-ng-agent
```

To install the Flume agent so Flume starts automatically on boot on Red Hat-compatible systems:

```
$ sudo yum install flume-ng-agent
```

To install the Flume agent so Flume starts automatically on boot on SLES systems:

```
$ sudo zypper install flume-ng-agent
```

To install the documentation:

To install the documentation on Ubuntu and other Debian systems:

```
$ sudo apt-get install flume-ng-doc
```

To install the documentation on Red Hat-compatible systems:

```
$ sudo yum install flume-ng-doc
```

To install the documentation on SLES systems:

```
$ sudo zypper install flume-ng-doc
```

Flume Configuration

Flume 1.x provides a template configuration file for `flume.conf` called `conf/flume-conf.properties.template` and a template for `flume-env.sh` called `conf/flume-env.sh.template`.

1. Copy the Flume template property file `conf/flume-conf.properties.template` to `conf/flume.conf`, then edit it as appropriate.

   ```
   $ sudo cp conf/flume-conf.properties.template conf/flume.conf
   ```

   This is where you define your sources, sinks, and channels, and the flow within an agent. By default, the properties file is configured to work out of the box using a sequence generator source, a logger sink, and a memory channel. For information on configuring agent flows in Flume 1.x, as well as more details about the supported sources, sinks and channels, see the documents listed under Viewing the Flume Documentation.

2. Optionally, copy the template flume-env.sh file `conf/flume-env.sh.template` to `conf/flume-env.sh`.

   ```
   $ sudo cp conf/flume-env.sh.template conf/flume-env.sh
   ```

   The `flume-ng` executable looks for a file named `flume-env.sh` in the `conf` directory, and sources it if it finds it. Some use cases for using `flume-env.sh` are to specify a bigger heap size for the flume agent, or to specify debugging or profiling options via JAVA_OPTS when developing your own custom Flume NG components such as sources and sinks. If you do not make any changes to this file, then you need not perform the copy as it is effectively empty by default.

Verifying the Installation

At this point, you should have everything necessary to run Flume, and the `flume-ng` command should be in your `$PATH`. You can test this by running:

```
$ flume-ng help
```

You should see something similar to this:

```
Usage: /usr/bin/flume-ng <command> [options]...

commands:
  help                      display this help text
  agent                     run a Flume agent
  avro-client               run an avro Flume client
  version                   show Flume version info

global options:
  --conf,-c <conf>          use configs in <conf> directory
  --classpath,-C <cp>       append to the classpath
```

```
  --dryrun,-d           do not actually start Flume, just print the command
  --Dproperty=value     sets a JDK system property value

agent options:
  --conf-file,-f <file> specify a config file (required)
  --name,-n <name>      the name of this agent (required)
  --help,-h             display help text

avro-client options:
  --rpcProps,-P <file>  RPC client properties file with server connection params
  --host,-H <host>      hostname to which events will be sent (required)
  --port,-p <port>      port of the avro source (required)
  --dirname <dir>       directory to stream to avro source
  --filename,-F <file>  text file to stream to avro source [default: std input]
  --headerFile,-R <file> headerFile containing headers as key/value pairs on each new
 line
  --help,-h             display help text

  Either --rpcProps or both --host and --port must be specified.

Note that if <conf> directory is specified, then it is always included first
in the classpath.
```

Note:

If Flume is not found and you installed Flume from a tarball, make sure that `$FLUME_HOME/bin` is in your `$PATH`.

Running Flume

If Flume is installed via an RPM or Debian package, you can use the following commands to start, stop, and restart the Flume agent via `init` scripts:

```
$ sudo service flume-ng-agent <start | stop | restart>
```

You can also run the agent in the foreground directly by using the `flume-ng agent` command:

```
$ /usr/bin/flume-ng agent -c <config-dir> -f <config-file> -n <agent-name>
```

For example:

```
$ /usr/bin/flume-ng agent -c /etc/flume-ng/conf -f /etc/flume-ng/conf/flume.conf -n
agent
```

Files Installed by the Flume RPM and Debian Packages

Resource	Location	Notes
Configuration Directory	`/etc/flume-ng/conf`	
Configuration File	`/etc/flume-ng/conf/flume.conf`	This configuration will be picked-up by the flume agent startup script.
Template of User Customizable Configuration File	`/etc/flume-ng/conf/flume-conf.properties.template`	Contains a sample config. To use this configuration you should copy this file onto `/etc/flume-ng/conf/flume.conf` and then modify as appropriate

Resource	Location	Notes
Template of User Customizable environment file	`/etc/flume-ng/conf/flume-env.sh.template`	If you want modify this file, copy it first and modify the copy
Daemon Log Directory	`/var/log/flume-ng`	Contains log files generated by flume agent
Default Flume Home	`/usr/lib/flume-ng`	Provided by RPMS and DEBS
Flume Agent startup script	`/etc/init.d/flume-ng-agent`	Provided by RPMS and DEBS
Recommended tar.gz Flume Home	`/usr/local/lib/flume-ng`	Recommended but installation dependent
Flume Wrapper Script	`/usr/bin/flume-ng`	Called by the Flume Agent startup script
Flume Agent configuration file	`/etc/default/flume-ng-agent`	Allows you to specify non-default values for the agent name and for the configuration file location

Supported Sources, Sinks, and Channels

The following tables list the only currently-supported sources, sinks, and channels. For more information, including information on developing custom components, see the documents listed under Viewing the Flume Documentation.

Sources

Type	Description	Implementation Class
avro	Avro Netty RPC event source. Listens on Avro port and receives events from external Avro streams.	AvroSource
netcat	Netcat style TCP event source. Listens on a given port and turns each line of text into an event.	NetcatSource
seq	Monotonically incrementing sequence generator event source	SequenceGeneratorSource
exec	Execute a long-lived Unix process and read from stdout.	ExecSource
syslogtcp	Reads syslog data and generates flume events. Creates a new event for a string of characters separated by carriage return (\n).	SyslogTcpSource
syslogudp	Reads syslog data and generates flume events. Treats an entire message as a single event.	SyslogUDPSource

Type	Description	Implementation Class
org.apache.flume.source.avroLegacy.AvroLegacySource	Allows the Flume 1.x agent to receive events from Flume 0.9.4 agents over avro rpc.	AvroLegacySource
org.apache.flume.source.thriftLegacy.ThriftLegacySource	Allows the Flume 1.x agent to receive events from Flume 0.9.4 agents over thrift rpc.	ThriftLegacySource
org.apache.flume.source.StressSource	Mainly for testing purposes. Not meant for production use. Serves as a continuous source of events where each event has the same payload.	StressSource
org.apache.flume.source.scribe.ScribeSource	Scribe event source. Listens on Scribe port and receives events from Scribe.	ScribeSource
multiport_syslogtcp	Multi-port capable version of the SyslogTcpSource.	MultiportSyslogTCPSource
spooldir	Used for ingesting data by placing files to be ingested into a "spooling" directory on disk.	SpoolDirectorySource
http	Accepts Flume events by HTTP POST and GET. GET should be used for experimentation only.	HTTPSource
org.apache.flume.source.jms.JMSSource	Reads messages from a JMS destination such as a queue or topic.	JMSSource
org.apache.flume.agent.embedded.EmbeddedSource	Used only by the Flume embedded agent. See Flume Developer Guide for more details.	EmbeddedSource
Other (custom)	You need to specify the fully-qualified name of the custom source, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's `lib` directory.	—

Sinks

Type	Description	Implementation Class
logger	Log events at INFO level via configured logging subsystem (log4j by default)	LoggerSink
avro	Sink that invokes a pre-defined Avro protocol method for all events it receives (when paired with an avro source, forms tiered collection)	AvroSink

Type	Description	Implementation Class
hdfs	Writes all events received to HDFS (with support for rolling, bucketing, HDFS-200 append, and more)	HDFSEventSink
file_roll	Writes all events received to one or more files.	RollingFileSink
org.apache.flume.hbase.HBaseSink	A simple sink that reads events from a channel and writes them synchronously to HBase. The AsyncHBaseSink is recommended.	HBaseSink
org.apache.flume.sink.hbase.AsyncHBaseSink	A simple sink that reads events from a channel and writes them asynchronously to HBase. This is the recommended HBase sink, but note that it does not support Kerberos.	AsyncHBaseSink
org.apache.flume.sink.solr.morphline.MorphlineSolrSink	Extracts and transforms data from Flume events, and loads it into Apache Solr servers. See the section on MorphlineSolrSink in the Flume User Guide listed under Viewing the Flume Documentation on page 168.	MorphlineSolrSink
Other (custom)	You need to specify the fully-qualified name of the custom sink, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's `lib` directory.	—

Channels

Type	Description	Implementation Class
memory	In-memory, fast, non-durable event transport	MemoryChannel
jdbc	JDBC-based, durable event transport (Derby-based)	JDBCChannel
file	File-based, durable event transport	FileChannel
Other (custom)	You need to specify the fully-qualified name of the custom channel, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's `lib` directory.	—

Providing for Disk Space Usage

It's important to provide plenty of disk space for any Flume File Channel. The largest consumers of disk space in the File Channel are the data logs. You can configure the File Channel to write these logs to multiple data directories. The following space will be consumed by default in each data directory:

- Current log file (up to 2 GB)
- Last log file (up to 2 GB)
- Pending delete log file (up to 2 GB)

Events in the queue could cause many more log files to be written, each of them up 2 GB in size by default.

You can configure both the maximum log file size (`MaxFileSize`) and the directories the logs will be written to (`DataDirs)` when you configure the File Channel; see the File Channel section of the Flume User Guide for details.

Using an On-disk Encrypted File Channel

Flume supports on-disk encryption of data on the local disk. To implement this:

- Generate an encryption key to use for the Flume Encrypted File Channel
- Configure on-disk encryption by setting parameters in the `flume.conf` file

Important:

Flume on-disk encryption operates with a maximum strength of 128-bit AES encryption unless the JCE unlimited encryption cryptography policy files are installed. Please see this Oracle document for information about enabling strong cryptography with JDK 1.6:
http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html

Consult your security organization for guidance on the acceptable strength of your encryption keys. Cloudera has tested with AES-128, AES-192, and AES-256.

Generating Encryption Keys

Use the `keytool` program included in Oracle JDK 1.6 to create the AES encryption keys for use with Flume.

The command to generate a 128-bit key that uses the same password as the key store password is:

```
keytool -genseckey -alias key-1 -keyalg AES -keysize 128 -validity 9000 \
-keystore test.keystore -storetype jceks \
-storepass keyStorePassword
```

The command to generate a 128-bit key that uses a different password from that used by the key store is:

```
keytool -genseckey -alias key-0 -keypass keyPassword -keyalg AES \
-keysize 128 -validity 9000 -keystore test.keystore \
-storetype jceks -storepass keyStorePassword
```

The key store and password files can be stored anywhere on the file system; both files should have `flume` as the owner and `0600` permissions.

Please note that `-keysize` controls the strength of the AES encryption key, in bits; 128, 192, and 256 are the allowed values.

Configuration

Flume on-disk encryption is enabled by setting parameters in the `/etc/flume-ng/conf/flume.conf` file.

Basic Configuration

The first example is a basic configuration with an alias called `key-0` that uses the same password as the key store:

```
agent.channels.ch-0.type = file
agent.channels.ch-0.capacity = 10000
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING
agent.channels.ch-0.encryption.activeKey = key-0
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0
```

In the next example, `key-0` uses its own password which may be different from the key store password:

```
agent.channels.ch-0.type = file
agent.channels.ch-0.capacity = 10000
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING
agent.channels.ch-0.encryption.activeKey = key-0
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0
agent.channels.ch-0.encryption.keyProvider.keys.key-0.passwordFile =
/path/to/key-0.password
```

Changing Encryption Keys Over Time

To modify the key, modify the configuration as shown below. This example shows how to change the configuration to use `key-1` instead of `key-0`:

```
agent.channels.ch-0.type = file
agent.channels.ch-0.capacity = 10000
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING
agent.channels.ch-0.encryption.activeKey = key-1
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0 key-1
```

The same scenario except that `key-0` and `key-1` have their own passwords is shown here:

```
agent.channels.ch-0.type = file
agent.channels.ch-0.capacity = 10000
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING
agent.channels.ch-0.encryption.activeKey = key-1
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0 key-1
agent.channels.ch-0.encryption.keyProvider.keys.key-0.passwordFile =
/path/to/key-0.password
agent.channels.ch-0.encryption.keyProvider.keys.key-1.passwordFile =
/path/to/key-1.password
```

Troubleshooting

If the unlimited strength JCE policy files are not installed, an error similar to the following is printed in the `flume.log`:

```
07 Sep 2012 23:22:42,232 ERROR [lifecycleSupervisor-1-0]
(org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.getCipher:137) -
Unable to load key using transformation: AES/CTR/NoPadding; Warning: Maximum allowed
key length = 128 with the available JCE security policy files. Have you installed the
 JCE unlimited strength jurisdiction policy files?
java.security.InvalidKeyException: Illegal key size
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.init(DashoA13*..)
at javax.crypto.Cipher.init(DashoA13*..)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.getCipher(AESCTRNoPaddingProvider.java:120)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.access$200(AESCTRNoPaddingProvider.java:35)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$AESCTRNoPaddingDecryptor.<init>(AESCTRNoPaddingProvider.java:94)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$AESCTRNoPaddingDecryptor.<init>(AESCTRNoPaddingProvider.java:91)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$DecryptorBuilder.build(AESCTRNoPaddingProvider.java:66)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$DecryptorBuilder.build(AESCTRNoPaddingProvider.java:62)
at
org.apache.flume.channel.file.encryption.CipherProviderFactory.getDecrypter(CipherProviderFactory.java:47)
at org.apache.flume.channel.file.LogFileV3$SequentialReader.<init>(LogFileV3.java:257)
at
org.apache.flume.channel.file.LogFileFactory.getSequentialReader(LogFileFactory.java:110)
at org.apache.flume.channel.file.ReplayHandler.replayLog(ReplayHandler.java:258)
at org.apache.flume.channel.file.Log.replay(Log.java:339)
at org.apache.flume.channel.file.FileChannel.start(FileChannel.java:260)
at
org.apache.flume.lifecycle.LifecycleSupervisor$MonitorRunnable.run(LifecycleSupervisor.java:236)
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:441)
at java.util.concurrent.FutureTask$Sync.innerRunAndReset(FutureTask.java:317)
at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:150)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$101(ScheduledThreadPoolExecutor.java:98)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.runPeriodic(ScheduledThreadPoolExecutor.java:180)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:204)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:662)
```

Viewing the Flume Documentation

For additional Flume documentation, see the Flume User Guide and the Flume Developer Guide.

For additional information about Flume, see the Apache Flume wiki.

HBase Installation

Apache HBase provides large-scale tabular storage for Hadoop using the Hadoop Distributed File System (HDFS). Cloudera recommends installing HBase in a standalone mode before you try to run it on a whole cluster.

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper/YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation and the instructions for upgrading to CDH 5 .

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Use the following sections to install, update, and configure HBase:

- New Features and Changes for HBase in CDH 5
- Upgrading HBase
- Installing HBase
- Configuration Settings
- Starting HBase in Standalone Mode
- Configuring HBase in Pseudo-Distributed Mode
- Deploying HBase in a Cluster
- Using the Hbase Shell
- Using MapReduce with HBase
- Troubleshooting
- Apache HBase Documentation
- HBase Replication
- Configuring Snapshots

New Features and Changes for HBase in CDH 5

CDH 5.0.x and 5.1.x each include major upgrades to HBase. Each of these upgrades provides exciting new features, as well as things to keep in mind when upgrading from a previous version.

For new features introduced in CDH 5.0.x, skip to CDH 5.0.x HBase Changes.

CDH 5.1 HBase Changes

CDH 5.1 introduces HBase 0.98, which represents a major upgrade to HBase. This upgrade introduces several new features, including a section of features which are considered experimental and should not be used in a production environment. This overview provides information about the most important features, how to use them, and where to find out more information. Cloudera appreciates your feedback about these features.

In addition to HBase 0.98, Cloudera has pulled in changes from HBASE-10883, HBASE-10964, HBASE-10823, HBASE-10916, and HBASE-11275. Implications of these changes are detailed below and in the Release Notes.

BucketCache Block Cache

A new offheap BlockCache implementation, BucketCache, was introduced as an experimental feature in CDH 5 Beta 1, and is now fully supported in CDH 5.1. BucketCache can be used in either of the following two configurations:

- As a CombinedBlockCache with both onheap and offheap caches.
- As an L2 cache for the default onheap LruBlockCache

BucketCache requires less garbage-collection than SlabCache, which is the other offheap cache implementation in HBase. It also has many optional configuration settings for fine-tuning. All available settings are documented in the API documentation for CombinedBlockCache. Following is a simple example configuration.

1. First, edit `hbase-env.sh` and set `-XX:MaxDirectMemorySize` to the total size of the desired onheap plus offheap, in this case, 5 GB (but expressed as `5G`). To edit the configuration, use an Advanced Configuration Snippet if you use Cloudera Manager, or edit the file directly otherwise.

   ```
   -XX:MaxDirectMemorySize=5G
   ```

2. Next, add the following configuration to `hbase-site.xml`. To edit the configuration, use an Advanced Configuration Snippet if you use Cloudera Manager, or edit the file directly otherwise. This configuration uses 80% of the `-XX:MaxDirectMemorySize` (4 GB) for offheap, and the remainder (1 GB) for onheap.

   ```
   <property>
     <name>hbase.bucketcache.ioengine</name>
     <value>offheap</value>
   </property>
   <property>
     <name>hbase.bucketcache.percentage.in.combinedcache</name>
     <value>0.8</value>
   </property>
   <property>
     <name>hbase.bucketcache.size</name>
     <value>5120</value>
   </property>
   ```

3. Restart or rolling restart your cluster for the configuration to take effect.

Access Control for EXEC Permissions

A new access control level has been added to check whether a given user has EXEC permission. This can be specified at the level of the cluster, table, row, or cell.

To use EXEC permissions, perform the following procedure.

- Install the AccessController coprocessor either as a system coprocessor or on a table as a table coprocessor.
- Set the `hbase.security.exec.permission.checks` configuration setting in `hbase-site.xml` to `true`. To edit the configuration, use an Advanced Configuration Snippet if you use Cloudera Manager, or edit the file directly otherwise.

For more information on setting and revoking security permissions, see the Access Control section of the *Apache HBase Reference Guide*.

Reverse Scan API

A reverse scan API has been introduced. This allows you to scan a table in reverse. Previously, if you wanted to be able to access your data in either direction, you needed to store the data in two separate tables, each ordered differently. This feature was implemented in HBASE-4811.

To use the reverse scan feature, use the new `Scan.setReversed(boolean reversed)` API. If you specify a startRow and stopRow, to scan in reverse, the startRow needs to be lexicographically after the stopRow. See the Scan API documentation for more information.

MapReduce Over Snapshots

You can now run a MapReduce job over a snapshot from HBase, rather than being limited to live data. This provides the ability to separate your client-side work load from your live cluster if you need to run resource-intensive MapReduce jobs and can tolerate using potentially-stale data. You can either run the MapReduce job on the snapshot within HBase, or export the snapshot and run the MapReduce job against the exported file.

Running a MapReduce job on an exported file outside of the scope of HBase relies on the permissions of the underlying filesystem and server, and bypasses ACLs, visibility labels, and encryption that may otherwise be provided by your HBase cluster.

A new API, TableSnapshotInputFormat, is provided. For more information, see TableSnapshotInputFormat.

MapReduce over snapshots was introduced in CDH 5.0.

Stateless Streaming Scanner over REST

A new stateless streaming scanner is available over the REST API. Using this scanner, clients do not need to restart a scan if the REST server experiences a transient failure. All query parameters are specified during the REST request. Query parameters include `startrow`, `endrow`, `columns`, `starttime`, `endtime`, `maxversions`, `batchtime`, and limit. Following are a few examples of using the stateless streaming scanner.

Scan the entire table, return the results in JSON.

```
curl -H "Accept: application/json" https://localhost:8080/ExampleScanner/*
```

Scan the entire table, return the results in XML.

```
curl -H "Content-Type: text/xml" https://localhost:8080/ExampleScanner/*
```

Scan only the first row.

```
curl -H "Content-Type: text/xml" \
https://localhost:8080/ExampleScanner/*?limit=1
```

Scan only specific columns.

```
curl -H "Content-Type: text/xml" \
https://localhost:8080/ExampleScanner/*?columns=a:1,b:1
```

Scan for rows between starttime and endtime.

```
curl -H "Content-Type: text/xml" \
https://localhost:8080/ExampleScanner/*?starttime=1389900769772\
&endtime=1389900800000
```

Scan for a given row prefix.

```
curl -H "Content-Type: text/xml" https://localhost:8080/ExampleScanner/test*
```

For full details about the stateless streaming scanner, see the API documentation for this feature.

Delete Methods of Put Class Now Use Constructor Timestamps

The `Delete()` methods of the `Put` class of the HBase Client API previously ignored the constructor's timestamp, and used the value of `HConstants.LATEST_TIMESTAMP`. This behavior was different from the behavior of the `add()` methods. The `Delete()` methods now use the timestamp from the constructor, creating consistency in behavior across the `Put` class. See HBASE-10964.

Experimental Features

- **Warning:** These features are still considered experimental. Experimental features are not supported and Cloudera does not recommend using them in production environments or with important data.

Visibility Labels

You can now use visibility labels, such as CONFIDENTIAL, TOPSECRET, and PUBLIC, at the cell level. In addition, these labels can be grouped using logical operators `&`, `|`, and `!` (AND, OR, NOT). A given user is associated with a set of visibility labels, and the policy for associating the labels is pluggable. The default plugin (`org.apache.hadoop.hbase.security.visibility.DefaultScanLabelGenerator`) checks for visibility labels on cells that would be returned by a Get or Scan and drops the cells that a user is not authorized to see. You can specify custom implementations of `ScanLabelGenerator` by setting the property `hbase.regionserver.scan.visibility.label.generator.class` to a comma-separated list of classes.

No labels are configured by default. You can add a label to the system using either the `VisibilityClient#addLabels()` API or the `add_label` shell command. Similar APIs and shell commands are provided for deleting labels and assigning them to users. Only a user with superuser access (the `hbase.superuser` access level) can perform these operations.

To assign a visibility label to a cell, you can label the cell using the API method `Mutation#setCellVisibility(new CellVisibility(<labelExp>));`.

Visibility labels and request authorizations cannot contain the symbols `&`, `|`, `!`, `(` and `)` because they are reserved for constructing visibility expressions. See HBASE-10883.

For more information about visibility labels, see the Visibility Labels section of the *Apache HBase Reference Guide*.

If you use visibility labels along with access controls, you must ensure that the Access Controller is loaded before the Visibility Controller in the list of coprocessors. This is the default configuration. See HBASE-11275.

In order to use per-cell access controls or visibility labels, you must use HFile version 3. To enable HFile version 3, add the following to `hbase-site.xml`, using an advanced configuration snippet if you use Cloudera Manager, or directly to the file if your deployment is unmanaged. Changes will take effect after the next major compaction.

```
<property>
  <name>hfile.format.version</name>
  <value>3</value>
</property>
```

Visibility labels are an **experimental** feature introduced in CDH 5.1.

Per-Cell Access Controls

You can now specify access control levels at the per-cell level, as well as at the level of the cluster, table, or row.

A new parent class has been provided, which encompasses `Get`, `Scan`, and `Query`. This change also moves the `getFilter` and `setFilter` methods of `Get` and `Scan` to the common parent class. Client code may need to be recompiled to take advantage of per-cell ACLs. See the Access Control section of the *Apache HBase Reference Guide* for more information.

The ACLS for cells having timestamps in the future are not considered for authorizing the pending mutation operations. See HBASE-10823.

If you use visibility labels along with access controls, you must ensure that the Access Controller is loaded before the Visibility Controller in the list of coprocessors. This is the default configuration.

In order to use per-cell access controls or visibility labels, you must use HFile version 3. To enable HFile version 3, add the following to `hbase-site.xml`, using an advanced configuration snippet if you use Cloudera Manager, or directly to the file if your deployment is unmanaged.. Changes will take effect after the next major compaction.

```
<property>
  <name>hfile.format.version</name>
  <value>3</value>
</property>
```

Per-cell access controls are an **experimental** feature introduced in CDH 5.1.

Transparent Server-Side Encryption

Transparent server-side encryption can now be enabled for both HFiles and write-ahead logs (WALs), to protect their contents at rest. To configure transparent encryption, first create an encryption key, then configure the appropriate settings in `hbase-site.xml`. See the Transparent Encryption section in the *Apache HBase Reference Guide* for more information.

Transparent server-side encryption is an **experimental** feature introduced in CDH 5.1.

Stripe Compaction

Stripe compaction is a compaction scheme that segregates the data inside a region by row key, creating "stripes" of data which are visible within the region but transparent to normal operations. This striping improves read performance in common scenarios and greatly reduces variability by avoiding large and/or inefficient compactions.

Configuration guidelines and more information are available at Stripe Compaction.

To configure stripe compaction for a single table from within the HBase shell, use the following syntax.

```
alter <table>, CONFIGURATION => {<setting> => <value>}
 Example: alter 'orders', CONFIGURATION => {'hbase.store.stripe.fixed.count' => 10}
```

To configure stripe compaction for a column family from within the HBase shell, use the following syntax.

```
alter <table>, {NAME => <column family>, CONFIGURATION => {<setting => <value>}}
 Example: alter 'logs', {NAME => 'blobs', CONFIGURATION =>
{'hbase.store.stripe.fixed.count' => 10}}
```

Stripe compaction is an **experimental** feature in CDH 5.1.

Distributed Log Replay

After a region server fails, its failed region is assigned to another region server, which is marked as "recovering" in ZooKeeper. A SplitLogWorker directly replays edits from the WAL of the failed region server to the region at its new location. When a region is in "recovering" state, it can accept writes but no reads (including Append and Increment), region splits or merges. Distributed Log Replay extends the distributed log splitting framework. It works by directly replaying WAL edits to another region server instead of creating `recovered.edits` files.

Distributed log replay provides the following advantages over using the current distributed log splitting functionality on its own.

- It eliminates the overhead of writing and reading a large number of `recovered.edits` files. It is not unusual for thousands of recovered.edits files to be created and written concurrently during a region server recovery. Many small random writes can degrade overall system performance.
- It allows writes even when a region is in recovering state. It only takes seconds for a recovering region to accept writes again.

To enable distributed log replay, set `hbase.master.distributed.log.replay` to `true` in `hbase-site.xml`. To edit the configuration, use an Advanced Configuration Snippet if you use Cloudera Manager, or edit the file directly otherwise.You must also enable HFile version 3. Distributed log replay is unsafe for rolling upgrades.

Distributed log replay is an **experimental** feature in CDH 5.1.

CDH 5.0.x HBase Changes

HBase in CDH 5.0.x is based on the Apache HBase 0.96 release. When upgrading to CDH 5.0.x, keep the following in mind.

Wire Compatibility

HBase in CDH 5.0.x (HBase 0.96) is not wire compatible with CDH 4 (based on 0.92 and 0.94 releases). Consequently, rolling upgrades from CDH 4 to CDH 5 are not possible because existing CDH 4 HBase clients cannot make requests to CDH 5 servers and CDH 5 HBase clients cannot make requests to CDH 4 servers. Clients of the Thrift and REST proxy servers, however, retain wire compatibility between CDH 4 and CDH 5.

Upgrade is Not Reversible

The upgrade from CDH 4 HBase to CDH 5 HBase is irreversible and requires HBase to be shut down completely. Executing the upgrade script reorganizes existing HBase data stored on HDFS into new directory structures, converts HBase 0.90 HFile v1 files to the improved and optimized HBase 0.96 HFile v2 file format, and rewrites the `hbase.version` file. This upgrade also removes transient data stored in ZooKeeper during the conversion to the new data format.

These changes were made to reduce the impact in future major upgrades. Previously HBase used brittle custom data formats and this move shifts HBase's RPC and persistent data to a more evolvable Protocol Buffer data format.

API Changes

The HBase User API (Get, Put, Result, Scanner etc; see Apache HBase API documentation) has evolved and attempts have been made to make sure the HBase Clients are source code compatible and thus should recompile without needing any source code modifications. This cannot be guaranteed however, since with the conversion to Protocol Buffers (ProtoBufs), some relatively obscure APIs have been removed. Rudimentary efforts have also been made to preserve recompile compatibility with advanced APIs such as Filters and Coprocessors. These advanced APIs are still evolving and our guarantees for API compatibility are weaker here.

For information about changes to custom filters, see Custom Filters.

As of 0.96, the User API has been marked and all attempts at compatibility in future versions will be made. A version of the javadoc that only contains the User API can be found here.

HBase Metrics Changes

HBase provides a metrics framework based on JMX beans. Between HBase 0.94 and 0.96, the metrics framework underwent many changes. Some beans were added and removed, some metrics were moved from one bean to another, and some metrics were renamed or removed. Click here to download the CSV spreadsheet which provides a mapping.

Custom Filters

If you used custom filters written for HBase 0.94, you need to recompile those filters forHBase 0.96. The custom filter must be altered to fit with the newer interface that uses protocol buffers. Specifically two new methods, `toByteArray(…)` and `parseFrom(…)`, which are detailed in detailed in the Filter API. These should be used instead of the old methods `write(…)` and `readFields(…)`, so that protocol buffer serialization is used. To see what changes were required to port one of HBase's own custom filters, see the Git commit that represented porting the `SingleColumnValueFilter` filter.

Checksums

In CDH 4, HBase relied on HDFS checksums to protect against data corruption. When you upgrade to CDH 5, HBase checksums are now turned on by default. With this configuration, HBase reads data and then verifies the checksums. Checksum verification inside HDFS will be switched off. If the HBase-checksum verification fails, then the HDFS checksums are used instead for verifying data that is being read from storage. Once you turn on HBase checksums, you will not be able to roll back to an earlier HBase version.

You should see a modest performance gain after setting `hbase.regionserver.checksum.verify` to true for data that is not already present in the Region Server's block cache.

To enable or disable checksums, modify the following configuration properties in `hbase-site.xml`. To edit the configuration, use an Advanced Configuration Snippet if you use Cloudera Manager, or edit the file directly otherwise.

```
<property>
  <name>hbase.regionserver.checksum.verify</name>
  <value>true</value>
  <description>
      If set to  true, HBase will read data and then verify checksums  for
      hfile blocks. Checksum verification inside HDFS will be switched off.
      If the hbase-checksum verification fails, then it will  switch back to
      using HDFS checksums.
  </description>
</property>
```

The default value for the `hbase.hstore.checksum.algorithm` property has also changed to `CRC32`. Previously, Cloudera advised setting it to `NULL` due to performance issues which are no longer a problem.

```
<property>
   <name>hbase.hstore.checksum.algorithm</name>
   <value>CRC32</value>
   <description>
     Name of an algorithm that is used to compute checksums. Possible values
     are NULL, CRC32, CRC32C.
   </description>
 </property>
```

Upgrading HBase

Note:

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

Important:

Before you start, make sure you have read and understood the previous section, New Features and Changes for HBase in CDH 5 on page 169, and check the Known Issues and Work Arounds in CDH 5 and Incompatible Changes for HBase .

Coprocessors and Custom JARs

When upgrading HBase from one major version to another (such as moving from CDH 4 to CDH 5), you must recompile coprocessors and custom JARs *after* the upgrade.

Upgrading HBase from CDH 4 to CDH 5

CDH 5.0 HBase is based on Apache HBase 0.96.1.1 Remember that once a cluster has been upgraded to CDH 5, it cannot be reverted to CDH 4. To ensure a smooth upgrade, this section guides you through the steps involved in upgrading HBase from the older CDH 4.x releases to CDH 5.

These instructions also apply to upgrading HBase from CDH 4.x directly to CDH 5.1.0, which is a supported path.

Prerequisites

HDFS and ZooKeeper should be available while upgrading HBase.

Overview of Upgrade Procedure

Before you can upgrade HBase from CDH 4 to CDH 5, your HFiles must be upgraded from HFile v1 format to HFile v2, because CDH 5 no longer supports HFile v1. The upgrade procedure itself is different if you are using Cloudera Manager or the command line, but has the same results. The first step is to check for instances of HFile v1 in the HFiles and mark them to be upgraded to HFile v2, and to check for and report about corrupted files or files with unknown versions, which need to be removed manually. The next step is to rewrite the HFiles during the next major compaction. After the HFiles are upgraded, you can continue the upgrade.

Upgrade HBase Using the Command Line

CDH 5 comes with an upgrade script for HBase. You can run `bin/hbase --upgrade` to see its Help section. The script runs in two modes: `-check` and `-execute`.

Step 1: Check for HFile v1 files and compact if necessary

1. Run the upgrade command in `-check` mode, and examine the output.

```
$ bin/hbase upgrade -check
```

Your output should be similar to the following:

```
Tables Processed:
hdfs://localhost:41020/myHBase/.META.
hdfs://localhost:41020/myHBase/usertable
hdfs://localhost:41020/myHBase/TestTable
hdfs://localhost:41020/myHBase/t

Count of HFileV1: 2
HFileV1:
hdfs://localhost:41020/myHBase/usertable
/fa02dac1f38d03577bd0f7e666f12812/family/249450144068442524
hdfs://localhost:41020/myHBase/usertable
/ecdd3eaee2d2fcf8184ac025555bb2af/family/249450144068442512

Count of corrupted files: 1
Corrupted Files:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812/family/1
Count of Regions with HFileV1: 2
Regions to Major Compact:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812
hdfs://localhost:41020/myHBase/usertable/ecdd3eaee2d2fcf8184ac025555bb2af
```

In the example above, you can see that the script has detected two HFile v1 files, one corrupt file and the regions to major compact.

By default, the script scans the root directory, as defined by `hbase.rootdir`. To scan a specific directory, use the `--dir` option. For example, the following command scans the `/myHBase/testTable` directory.

```
bin/hbase upgrade --check --dir /myHBase/testTable
```

2. Trigger a major compaction on each of the reported regions. This major compaction rewrites the files from HFile v1 to HFile v2 format. To run the major compaction, start HBase Shell and issue the `major_compact` command.

```
$ bin/hbase shell
hbase> major_compact 'usertable'
```

You can also do this in a single step by using the echo shell built-in command.

```
$ echo "major_compact 'usertable'" | bin/hbase shell
```

3. Once all the HFileV1 files have been rewritten, running the upgrade script with the `-check` option again will return a "No HFile v1 found" message. It is then safe to proceed with the upgrade.

Step 2: Gracefully shutdown CDH 4 HBase cluster

Shutdown your CDH 4 HBase cluster before you run the upgrade script in `-execute` mode.

To shutdown HBase gracefully:

1. Stop the REST and Thrift server and clients, then stop the cluster.
   a. Stop the Thrift server and clients:

      ```
      sudo service hbase-thrift stop
      ```

      Stop the REST server:

      ```
      sudo service hbase-rest stop
      ```

   b. Stop the cluster by shutting down the master and the region servers:
      a. Use the following command on the master node:

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

      b. Use the following command on each node hosting a region server:

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

2. Stop the ZooKeeper Server:

   ```
   $ sudo service zookeeper-server stop
   ```

Step 4: Run the HBase upgrade script in -execute mode

Important: Before you proceed with Step 4, upgrade your CDH 4 cluster to CDH 5. See Upgrading to CDH 5 on page 58 for instructions.

This step executes the actual upgrade process. It has a verification step which checks whether or not the Master, RegionServer and backup Master znodes have expired. If not, the upgrade is aborted. This ensures no upgrade occurs while an HBase process is still running. If your upgrade is aborted even after shutting down the HBase cluster, retry after some time to let the znodes expire. Default znode expiry time is 300 seconds.

As mentioned earlier, ZooKeeper and HDFS should be available. If ZooKeeper is managed by HBase, then use the following command to start ZooKeeper.

```
./hbase/bin/hbase-daemon.sh start zookeeper
```

The upgrade involves three steps:

- **Upgrade Namespace:** This step upgrades the directory layout of HBase files.
- **Upgrade Znodes:** This step upgrades `/hbase/replication` (znodes corresponding to peers, log queues and so on) and `table` znodes (keep table enable/disable information). It deletes other znodes.
- **Log Splitting:** In case the shutdown was not clean, there might be some Write Ahead Logs (WALs) to split. This step does the log splitting of such WAL files. It is executed in a "non distributed mode", which could make the upgrade process longer in case there are too many logs to split. To expedite the upgrade, ensure you have completed a clean shutdown.

Run the upgrade command in `-execute` mode.

```
$ bin/hbase upgrade -execute
```

Your output should be similar to the following:

```
Starting Namespace upgrade
Created version file at hdfs://localhost:41020/myHBase with version=7
Migrating table testTable to hdfs://localhost:41020/myHBase/.data/default/testTable
…..
Created version file at hdfs://localhost:41020/myHBase with version=8
Successfully completed NameSpace upgrade.
Starting Znode upgrade
….
Successfully completed Znode upgrade
Starting Log splitting
…
Successfully completed Log splitting
```

The output of the `-execute` command can either return a success message as in the example above, or, in case of a clean shutdown where no log splitting is required, the command would return a "No log directories to split, returning" message. Either of those messages indicates your upgrade was successful.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

Step 5 (Optional): Move Tables to Namespaces

CDH 5 introduces namespaces for HBase tables. As a result of the upgrade, all tables are automatically assigned to namespaces. The `root`, `meta`, and `acl` tables are added to the `hbase` system namespace. All other tables are assigned to the `default` namespace.

To move a table to a different namespace, take a snapshot of the table and clone it to the new namespace. After the upgrade, do the snapshot and clone operations before turning the modified application back on.

Warning: Do not move datafiles manually, as this can cause data corruption that requires manual intervention to fix.

Step 6: Recompile custom coprocessors and JARs.

Recompile any coprocessors and custom JARs, so that they will work with the new version of HBase.

FAQ

In order to prevent upgrade failures because of unexpired znodes, is there a way to check/force this before an upgrade?

The upgrade script "executes" the upgrade when it is run with the `-execute` option. As part of the first step, it checks for any live HBase processes (RegionServer, Master and backup Master), by looking at their znodes. If any such znode is still up, it aborts the upgrade and prompts the user to stop such processes, and wait until their znodes have expired. This can be considered an inbuilt check.

The `-check` option has a different use case: To check for HFile v1 files. This option is to be run on live CDH 4 clusters to detect HFile v1 and major compact any regions with such files.

What are the steps for Cloudera Manager to do the upgrade?

See Upgrading CDH in a Cloudera Manager Deployment in the Cloudera Manager documentation.

Upgrading HBase from an Earlier CDH 5 Release

Important: Rolling upgrade is not supported between a CDH 5 Beta release and this CDH 5 GA release. Cloudera recommends using Cloudera Manager if you need to do rolling upgrades.

To upgrade HBase from an earlier CDH 5 release, proceed as follows.

The instructions that follow assume that you are upgrading HBase as part of an upgrade to the latest CDH 5 release, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

Step 1: Perform a Graceful Cluster Shutdown

Note:

Upgrading via rolling restart is not supported.

To shut HBase down gracefully:

1. Stop the Thrift server and clients, then stop the cluster.
   a. Stop the Thrift server and clients:
      ```
      sudo service hbase-thrift stop
      ```
   b. Stop the cluster by shutting down the master and the region servers:
      a. Use the following command on the master node:
         ```
         sudo service hbase-master stop
         ```
      b. Use the following command on each node hosting a region server:
         ```
         sudo service hbase-regionserver stop
         ```
2. Stop the ZooKeeper Server:
   ```
   $ sudo service zookeeper-server stop
   ```

Step 2: Install the new version of HBase

Note:

You may want to take this opportunity to upgrade ZooKeeper, but you do not *have* to upgrade Zookeeper before upgrading HBase; the new version of HBase will run with the older version of Zookeeper. For instructions on upgrading ZooKeeper, see Upgrading ZooKeeper from an Earlier CDH 5 Release on page 360.

It is a good idea to back up the `/hbase` znode before proceeding. By default, this is in `/var/lib/zookeeper`.

To install the new version of HBase, follow directions in the next section, Installing HBase on page 180.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Installing HBase

To install HBase On Red Hat-compatible systems:

```
$ sudo yum install hbase
```

To install HBase on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase
```

To install HBase on SLES systems:

```
$ sudo zypper install hbase
```

Note: See also Starting HBase in Standalone Mode on page 183, Configuring HBase in Pseudo-Distributed Mode on page 185, and Deploying HBase on a Cluster on page 187 for more information on configuring HBase for different modes.

To list the installed files on Ubuntu and Debian systems:

```
$ dpkg -L hbase
```

To list the installed files on Red Hat and SLES systems:

```
$ rpm -ql hbase
```

You can see that the HBase package has been configured to conform to the Linux Filesystem Hierarchy Standard. (To learn more, run `man hier`).

You are now ready to enable the server daemons you want to use with Hadoop. You can also enable Java-based client access by adding the JAR files in `/usr/lib/hbase/` and `/usr/lib/hbase/lib/` to your Java class path.

Configuration Settings for HBase

This section contains information on configuring the Linux host and HDFS for HBase.

Using DNS with HBase

HBase uses the local hostname to report its IP address. Both forward and reverse DNS resolving should work. If your server has multiple interfaces, HBase uses the interface that the primary hostname resolves to. If this is insufficient, you can set `hbase.regionserver.dns.interface` in the `hbase-site.xml` file to indicate the primary interface. To work properly, this setting requires that your cluster configuration is consistent and every host has the same network interface configuration. As an alternative, you can set `hbase.regionserver.dns.nameserver` in the `hbase-site.xml` file to use a different DNS name server than the system-wide default.

Using the Network Time Protocol (NTP) with HBase

The clocks on cluster members must be synchronized for your cluster to function correctly. Some skew is tolerable, but excessive skew could generate odd behaviors. Run NTP or another clock synchrnoization mechanism on your cluster. If you experience problems querying data or unusual cluster operations, verify the system time. For more information about NTP, see the NTP website.

Setting User Limits for HBase

Because HBase is a database, it opens many files at the same time. The default setting of 1024 for the maximum number of open files on most Unix-like systems is insufficient. Any significant amount of loading will result in failures and cause error message such as `java.io.IOException...(Too many open files)` to be logged in the HBase or HDFS log files. For more information about this issue, see the Apache HBase Book. You may also notice errors such as:

```
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Exception
increateBlockOutputStream java.io.EOFException
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Abandoning block
blk_-6935524980745310745_1391901
```

Another setting you should configure is the number of processes a user is permitted to start. The default number of processes is typically 1024. Consider raising this value if you experience `OutOfMemoryException` errors.

Configuring ulimit for HBase

Cloudera recommends increasing the maximum number of file handles to more than 10,000. Note that increasing the file handles for the user who is running the HBase process is an operating system configuration, not an HBase configuration. Also, a common mistake is to increase the number of file handles for a particular user but, for whatever reason, HBase will be running as a different user. HBase prints the ulimit it is using on the first line in the logs. Make sure that it is correct.

To change the maximum number of open files for a given user, use the `ulimit -n` command while logged in as that user. To set the maximum number of processes a user may start, use the `ulimit -u` command. The `ulimit` command can be used to set many other limits besides the number of open files. Refer to the online documentation for your operating system, or the output of the `man ulimit` command, for more information. To make the changes persistent, you can add the command to the user's Bash initialization file (typically `~/.bash_profile` or `~/.bashrc`). Alternatively, you can configure the settings in the `Pluggable Authentication Module (PAM)` configuration files if your operating system uses PAM and includes the `pam_limits.so` shared library.

Configuring ulimit via Pluggable Authentication Modules

If you are using `ulimit`, you must make the following configuration changes:

1. In the `/etc/security/limits.conf` file, add the following lines, adjusting the values as appropriate. This assumes that your HDFS user is called `hdfs` and your HBase user is called `hbase`.

```
hdfs  -       nofile  32768
hdfs  -       nproc   2048
hbase -       nofile  32768
hbase -       nproc   2048
```

Note:

- Only the root user can edit this file.
- If this change does not take effect, check other configuration files in the /etc/security/limits.d/ directory for lines containing the hdfs or hbase user and the nofile value. Such entries may be overriding the entries in /etc/security/limits.conf.

To apply the changes in /etc/security/limits.conf on Ubuntu and Debian systems, add the following line in the /etc/pam.d/common-session file:

```
session required  pam_limits.so
```

For more information on the ulimit command or per-user operating system limits, refer to the documentation for your operating system.

Using dfs.datanode.max.transfer.threads with HBase

A Hadoop HDFS DataNode has an upper bound on the number of files that it can serve at any one time. The upper bound is controlled by the dfs.datanode.max.transfer.threads property (the property is spelled in the code exactly as shown here). Before loading, make sure you have configured the value for dfs.datanode.max.transfer.threads in the conf/hdfs-site.xml file (by default found in /etc/hadoop/conf/hdfs-site.xml) to at least 4096 as shown below:

```
<property>
  <name>dfs.datanode.max.transfer.threads</name>
  <value>4096</value>
</property>
```

Restart HDFS after changing the value for dfs.datanode.max.transfer.threads. If the value is not set to an appropriate value, strange failures can occur and an error message about exceeding the number of transfer threads will be added to the DataNode logs. Other error messages about missing blocks are also logged, such as:

```
06/12/14 20:10:31 INFO hdfs.DFSClient: Could not obtain block
blk_XXXXXXXXXXXXXXXXXXXXXX_YYYYYYYY from any node:
java.io.IOException: No live nodes contain current block. Will get new block locations
 from namenode and retry...
```

Note: The property dfs.datanode.max.transfer.threads is a HDFS 2 property which replaces the deprecated property dfs.datanode.max.xcievers.

Configuring BucketCache in HBase

The default BlockCache implementation in HBase is SlabCache. CDH 5 introduces support for BucketCache, though SlabCache is still used by default. For information about choosing and configuring the appropriate BlockCache implementation, refer to the API documentation for CacheConfig, as well as the BlockCache section of the Apache HBase Reference Guide.

Checksums in CDH 5

The default values for checksums have changed during the history of CDH 5. For information about configuring checksums, see New Features and Changes for HBase in CDH 5 on page 169.

Starting HBase in Standalone Mode

Note:

You can skip this section if you are already running HBase in distributed or pseudo-distributed mode.

By default, HBase ships configured for *standalone mode*. In this mode of operation, a single JVM hosts the HBase Master, an HBase Region Server, and a ZooKeeper quorum peer. HBase stores your data in a location on the local filesystem, rather than using HDFS. Standalone mode is only appropriate for initial testing.

Important:

If you have configured High Availability for the NameNode (HA), you cannot deploy HBase in standalone mode without modifying the default configuration, because both the standalone HBase process and ZooKeeper (required by HA) will try to bind to port 2181. You can configure a different port for ZooKeeper, but in most cases it makes more sense to deploy HBase in distributed mode in an HA cluster.

In order to run HBase in standalone mode, you must install the HBase Master package.

Installing the HBase Master

To install the HBase Master on Red Hat-compatible systems:

```
$ sudo yum install hbase-master
```

To install the HBase Master on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase-master
```

To install the HBase Master on SLES systems:

```
$ sudo zypper install hbase-master
```

Starting the HBase Master

- On Red Hat and SLES systems (using `.rpm` packages) you can now start the HBase Master by using the included service script:

```
$ sudo service hbase-master start
```

- On Ubuntu systems (using Debian packages) the HBase Master starts when the HBase package is installed.

To verify that the standalone installation is operational, visit `http://localhost:60010`. The list of Region Servers at the bottom of the page should include one entry for your local machine.

Note:

Although you have only started the master process, in *standalone* mode this same process is also internally running a region server and a ZooKeeper peer. In the next section, you will break out these components into separate JVMs.

If you see this message when you start the HBase standalone master:

```
Starting Hadoop HBase master daemon: starting master, logging to
/usr/lib/hbase/logs/hbase-hbase-master/cloudera-vm.out
Couldnt start ZK at requested address of 2181, instead got: 2182.  Aborting. Why?
Because clients (eg shell) wont be able to find this ZK quorum
hbase-master.
```

you will need to stop the hadoop-zookeeper-server (or zookeeper-server) or uninstall the hadoop-zookeeper-server (or zookeeper) package.

See also Accessing HBase by using the HBase Shell, Using MapReduce with HBase and Troubleshooting.

Installing and Starting the HBase Thrift Server

To install Thrift on Red Hat-compatible systems:

```
$ sudo yum install hbase-thrift
```

To install Thrift on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase-thrift
```

To install Thrift on SLES systems:

```
$ sudo zypper install hbase-thrift
```

You can now use the `service` command to start the Thrift server:

```
$ sudo service hbase-thrift start
```

Installing and Configuring HBase REST

To install HBase REST on Red Hat-compatible systems:

```
$ sudo yum install hbase-rest
```

To install HBase REST on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase-rest
```

To install HBase REST on SLES systems:

```
$ sudo zypper install hbase-rest
```

You can use the `service` command to run an `init.d` script, `/etc/init.d/hbase-rest`, to start the REST server; for example:

```
$ sudo service hbase-rest start
```

The script starts the server by default on port 8080. This is a commonly used port and so may conflict with other applications running on the same host.

If you need change the port for the REST server, configure it in `hbase-site.xml`, for example:

```
<property>
  <name>hbase.rest.port</name>
  <value>60050</value>
</property>
```

Note:

You can use HBASE_REST_OPTS in hbase-env.sh to pass other settings (such as heap size and GC parameters) to the REST server JVM.

Configuring HBase in Pseudo-Distributed Mode

Note: You can skip this section if you are already running HBase in distributed mode, or if you intend to use only standalone mode.

Pseudo-distributed mode differs from *standalone* mode in that each of the component processes run in a separate JVM. It differs from *distributed mode* in that each of the separate processes run on the same server, rather than multiple servers in a cluster. This section also assumes you wish to store your HBase data in HDFS rather than on the local filesystem.

Note: Before you start

- This section assumes you have already installed the HBase master and gone through the standalone configuration steps.
- If the HBase master is already running in standalone mode, stop it as follows before continuing with pseudo-distributed configuration:
- To stop the CDH 4 version: `sudo service hadoop-hbase-master stop`, *or*
- To stop the CDH 5 version if that version is already running: `sudo service hbase-master stop`

Modifying the HBase Configuration

To enable pseudo-distributed mode, you must first make some configuration changes. Open `/etc/hbase/conf/hbase-site.xml` in your editor of choice, and insert the following XML properties between the `<configuration>` and `</configuration>` tags. The `hbase.cluster.distributed` property directs HBase to start each process in a separate JVM. The `hbase.rootdir` property directs HBase to store its data in an HDFS filesystem, rather than the local filesystem. Be sure to replace `myhost` with the hostname of your HDFS NameNode (as specified by `fs.default.name` or `fs.defaultFS` in your `conf/core-site.xml` file); you may also need to change the port number from the default (8020).

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://myhost:8020/hbase</value>
</property>
```

Creating the /hbase Directory in HDFS

Before starting the HBase Master, you need to create the `/hbase` directory in HDFS. The HBase master runs as `hbase:hbase` so it does not have the required permissions to create a top level directory.

To create the /hbase directory in HDFS:

```
$ sudo -u hdfs hadoop fs -mkdir /hbase
$ sudo -u hdfs hadoop fs -chown hbase /hbase
```

> **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>`

Enabling Servers for Pseudo-distributed Operation

After you have configured HBase, you must enable the various servers that make up a distributed HBase cluster. HBase uses three required types of servers:

- Installing and Starting ZooKeeper Server
- Starting the HBase Master
- Starting an HBase RegionServer

Installing and Starting ZooKeeper Server

HBase uses ZooKeeper Server as a highly available, central location for cluster management. For example, it allows clients to locate the servers, and ensures that only one master is active at a time. For a small cluster, running a ZooKeeper node collocated with the NameNode is recommended. For larger clusters, contact Cloudera Support for configuration help.

Install and start the ZooKeeper Server in standalone mode by running the commands shown in the Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server on page 362

Starting the HBase Master

After ZooKeeper is running, you can start the HBase master in standalone mode.

```
$ sudo service hbase-master start
```

Starting an HBase RegionServer

The RegionServer is the part of HBase that actually hosts data and processes requests. The region server typically runs on all of the slave nodes in a cluster, but not the master node.

To enable the HBase RegionServer On Red Hat-compatible systems:

```
$ sudo yum install hbase-regionserver
```

To enable the HBase RegionServer on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase-regionserver
```

To enable the HBase RegionServer on SLES systems:

```
$ sudo zypper install hbase-regionserver
```

To start the RegionServer:

```
$ sudo service hbase-regionserver start
```

Verifying the Pseudo-Distributed Operation

After you have started ZooKeeper, the Master, and a RegionServer, the pseudo-distributed cluster should be up and running. You can verify that each of the daemons is running using the `jps` tool from the Oracle JDK, which you can obtain from here. If you are running a pseudo-distributed HDFS installation and a pseudo-distributed HBase installation on one machine, `jps` will show the following output:

```
$ sudo jps
32694 Jps
30674 HRegionServer
29496 HMaster
28781 DataNode
28422 NameNode
30348 QuorumPeerMain
```

You should also be able to navigate to `http://localhost:60010` and verify that the local region server has registered with the master.

Installing and Starting the HBase Thrift Server

The HBase Thrift Server is an alternative gateway for accessing the HBase server. Thrift mirrors most of the HBase client APIs while enabling popular programming languages to interact with HBase. The Thrift Server is multiplatform and more performant than REST in many situations. Thrift can be run collocated along with the region servers, but should not be collocated with the NameNode or the JobTracker. For more information about Thrift, visit http://incubator.apache.org/thrift/.

To enable the HBase Thrift Server On Red Hat-compatible systems:

```
$ sudo yum install hbase-thrift
```

To enable the HBase Thrift Server on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase-thrift
```

To enable the HBase Thrift Server on SLES systems:

```
$ sudo zypper install hbase-thrift
```

To start the Thrift server:

```
$ sudo service hbase-thrift start
```

See also Accessing HBase by using the HBase Shell on page 188, Using MapReduce with HBase on page 189 and Troubleshooting on page 189.

Deploying HBase on a Cluster

After you have HBase running in pseudo-distributed mode, the same configuration can be extended to running on a distributed cluster.

Note: Before you start

This section assumes that you have already installed the HBase Master and the HBase Region Server and gone through the steps for standalone and pseudo-distributed configuration. You are now about to distribute the processes across multiple hosts; see Choosing Where to Deploy the Processes on page 188.

Choosing Where to Deploy the Processes

For small clusters, Cloudera recommends designating one node in your cluster as the master node. On this node, you will typically run the HBase Master and a ZooKeeper quorum peer. These master processes may be collocated with the Hadoop NameNode and JobTracker for small clusters.

Designate the remaining nodes as slave nodes. On each node, Cloudera recommends running a Region Server, which may be collocated with a Hadoop TaskTracker (MRv1) and a DataNode. When co-locating with TaskTrackers, be sure that the resources of the machine are not oversubscribed – it's safest to start with a small number of MapReduce slots and work up slowly.

Configuring for Distributed Operation

After you have decided which machines will run each process, you can edit the configuration so that the nodes can locate each other. In order to do so, you should make sure that the configuration files are synchronized across the cluster. Cloudera strongly recommends the use of a configuration management system to synchronize the configuration files, though you can use a simpler solution such as `rsync` to get started quickly.

The only configuration change necessary to move from pseudo-distributed operation to fully-distributed operation is the addition of the ZooKeeper Quorum address in `hbase-site.xml`. Insert the following XML property to configure the nodes with the address of the node where the ZooKeeper quorum peer is running:

```
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>mymasternode</value>
</property>
```

The `hbase.zookeeper.quorum` property is a comma-separated list of hosts on which ZooKeeper servers are running. If one of the ZooKeeper servers is down, HBase will use another from the list. By default, the ZooKeeper service is bound to port 2181. To change the port, add the `hbase.zookeeper.property.clientPort` property to `hbase-site.xml` and set the value to the port you want ZooKeeper to use. For more information, see this chapter of the Apache HBase Reference Guide.

To start the cluster, start the services in the following order:

1. The ZooKeeper Quorum Peer
2. The HBase Master
3. Each of the HBase RegionServers

After the cluster is fully started, you can view the HBase Master web interface on port 60010 and verify that each of the slave nodes has registered properly with the master.

See also Accessing HBase by using the HBase Shell on page 188, Using MapReduce with HBase on page 189 and Troubleshooting on page 189. For instructions on improving the performance of local reads, see Tips and Guidelines on page 142.

Accessing HBase by using the HBase Shell

After you have started HBase, you can access the database by using the HBase Shell:

```
$ hbase shell
```

Using MapReduce with HBase

To run MapReduce jobs that use HBase, you need to add the HBase and Zookeeper JAR files to the Hadoop Java classpath. You can do this by adding the following statement to each job:

```
TableMapReduceUtil.addDependencyJars(job);
```

This distributes the JAR files to the cluster along with your job and adds them to the job's classpath, so that you do not need to edit the MapReduce configuration.

You can find more information about `addDependencyJars` in the documentation listed under Viewing the HBase Documentation on page 190.

When getting an `Configuration` object for a HBase MapReduce job, instantiate it using the `HBaseConfiguration.create()` method.

Troubleshooting

The Cloudera HBase packages have been configured to place logs in `/var/log/hbase`. Cloudera recommends tailing the `.log` files in this directory when you start HBase to check for any error messages or failures.

Table Creation Fails after Installing LZO

If you install LZO after starting the Region Server, you will not be able to create a table with LZO compression until you re-start the Region Server.

Why this happens

When the Region Server starts, it runs CompressionTest and caches the results. When you try to create a table with a given form of compression, it refers to those results. You have installed LZO since starting the Region Server, so the cached results, which pre-date LZO, cause the create to fail.

What to do

Restart the Region Server. Now table creation with LZO will succeed.

Thrift Server Crashes after Receiving Invalid Data

The Thrift server may crash if it receives a large amount of invalid data, due to a buffer overrun.

Why this happens

The Thrift server allocates memory to check the validity of data it receives. If it receives a large amount of invalid data, it may need to allocate more memory than is available. This is due to a limitation in the Thrift library itself.

What to do

To prevent the possibility of crashes due to buffer overruns, use the framed and compact transport protocols. These protocols are disabled by default, because they may require changes to your client code. The two options to add to your hbase-site.xml are `hbase.regionserver.thrift.framed` and `hbase.regionserver.thrift.compact`. Set each of these to `true`, as in the XML below. You can also specify the maximum frame size, using the `hbase.regionserver.thrift.framed.max_frame_size_in_mb` option.

```
<property>
  <name>hbase.regionserver.thrift.framed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.regionserver.thrift.framed.max_frame_size_in_mb</name>
  <value>2</value>
</property>
```

```
<property>
  <name>hbase.regionserver.thrift.compact</name>
  <value>true</value>
</property>
```

Viewing the HBase Documentation

For additional HBase documentation, see http://archive.cloudera.com/cdh5/cdh/5/hbase/.

HBase Replication

HBase replication provides a means of copying the data from one HBase cluster to another (typically distant) HBase cluster. It is designed for data recovery rather than failover.

The cluster receiving the data from user applications is called the **master** cluster, and the cluster receiving the replicated data from the master is called the **slave** cluster.

Types of Replication

You can implement any of the following replication models:

- Master-slave replication
- Master-master replication
- Cyclic replication

In all cases, the principle of replication is similar to that of MySQL master/slave replication in which each transaction on the master cluster is replayed on the slave cluster. In the case of HBase, the Write-Ahead Log (WAL) or HLog records all the transactions (Put/Delete) and the master cluster Region Servers ship the edits to the slave cluster Region Servers. This is done asynchronously, so having the slave cluster in a distant data center does not cause high latency at the master cluster.

Master-Slave Replication

This is the basic replication model, in which transactions on the master cluster are replayed on the slave cluster, as described above. For instructions on configuring master-slave replications, see Deploying HBase Replication.

Master-Master Replication

In this case, the slave cluster in one relationship can act as the master in a second relationship, and the slave in the second relationship can act as master in a third relationship, and so on.

Cyclic Replication

In the cyclic replication model, the slave cluster acts as master cluster for the original master. This sort of replication is useful when both the clusters are receiving data from different sources and you want each of these clusters to have the same data.

Important:

The normal configuration for cyclic replication is two clusters; you can configure more, but if you do, loop detection is not guaranteed in every case. Follow these guidelines.

Points to Note about Replication

- You make the configuration changes on the master cluster side.

- In the case of master-master replication, you make the changes on both sides.
- Replication works at the table-column-family level. The family should exist on all the slaves. (You can have additional, non replicating families on both sides).
- 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 master-master case, this may be undesirable, creating identical counters that overwrite one another. (See https://issues.apache.org/jira/browse/HBase-2804.)
- Make sure the master and slave clusters are time-synchronized with each other. Cloudera recommends you use Network Time Protocol (NTP).

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.*x* or CDH 5.*x*.
- Every table that contains families that are scoped for replication must exist on each cluster and have exactly the same name.
- HBase version 0.92 or greater is required for multiple slaves, master-master, or cyclic replication..

Deploying HBase Replication

Follow these steps to enable replication from one cluster to another.

1. Edit `${HBASE_HOME}/conf/hbase-site.xml` on both clusters and add the following:

```
<property>
   <name>hbase.replication</name>
   <value>true</value>
</property>
```

2. Push `hbase-site.xml` to all nodes.
3. Restart HBase if it was running.
4. Run the following command in the HBase master's shell while it's running:

```
add_peer
```

This will show you the help for setting up the replication stream between the clusters. The command takes the form:

```
add_peer '<n>',
"slave.zookeeper.quorum:zookeeper.clientport.:zookeeper.znode.parent"
```

where <n> is the peer ID; it should not be more than two characters (longer IDs may work, but have not been tested).

Example:

```
hbase> add_peer '1', "zk.server.com:2181:/hbase"
```

Note:

If both clusters use the same Zookeeper cluster, you need to use a different `zookeeper.znode.parent` for each so that they don't write to the same directory.

5. Once you have a peer, enable replication on your column families. One way to do this is to alter the table and set the scope like this:

   ```
   disable 'your_table'
   alter 'your_table', {NAME => 'family_name', REPLICATION_SCOPE => '1'}
   enable 'your_table'
   ```

   Currently, a scope of 0 (default) means that the data will not be replicated and a scope of 1 means that it will. This could change in the future.

6. To list all configured peers, run the following command in the master's shell:

   ```
   list_peers
   ```

You can confirm that your setup works by looking at any Region Server's log on the master cluster; look for the lines such as the following:

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 170.22.64.15:62020
```

This indicates that one Region Server from the slave cluster was chosen for replication.

Deploying Master-Master or Cyclic Replication

For master-master or cyclic replication, repeat the above steps on each master cluster: add the `hbase.replication` property and set it to `true`, push the resulting `hbase-site.xml` to all nodes of this master cluster, use add_peer to add the slave cluster, and enable replication on the column families.

Guidelines for Replication across Three or More Clusters

When configuring replication among three or more clusters, follow these guidelines:

1. The replication relationships should have *either*:
   - **No loops** (as in A->B->C, or A->B->C->D, etc.), *or*
   - If a loop exists, it should be a **complete cycle** (as in A->B->C->A, or A->B->C->D->A, etc.)
2. Cloudera recommends you enable `KEEP_DELETED_CELLS` on column families in the slave cluster, where `REPLICATION_SCOPE=1`in the master cluster; for example:
   - On the master:

     ```
     create 't1',{NAME=>'f1', REPLICATION_SCOPE=>1}
     ```
   - On the slave:

     ```
     create 't1',{NAME=>'f1', KEEP_DELETED_CELLS=>'true'}
     ```

Disabling Replication at the Peer Level

Use the command `disable_peer ("<peerID>")` to disable replication for a specific peer. This will stop replication to the peer, but the logs will be kept for future reference.

To re-enable the peer, use the command `enable_peer(<"peerID">)`. Replication resumes where it was stopped.

Examples:

- To disable peer 1:

```
disable_peer("1")
```

- To re-enable peer 1:

```
enable_peer("1")
```

Stopping Replication in an Emergency

If replication is causing serious problems, you can stop it while the clusters are running.

- **Warning:**

  Do this only in case of a serious problem; it may cause data loss.

To stop replication in an emergency:

Open the shell on the master cluster and use the `stop_replication` command. For example:

```
hbase(main):001:0> stop_replication
```

Already queued edits will be replicated after you use the `stop_replication` command, but new entries will not.

To start replication again, use the `start_replication` command.

Initiating Replication of Pre-existing Data

You may need to start replication from some point in the past. For example, suppose you have a primary HBase cluster in one location and are setting up a disaster-recovery (DR) cluster in another. To initialize the DR cluster, you need to copy over the existing data from the primary to the DR cluster, so that when you need to switch to the DR cluster you have a full copy of the data generated by the primary cluster. Once that is done, replication of new data can proceed as normal.

To start replication from an earlier point in time, run a `copyTable` command (defining the start and end timestamps), while enabling replication. Proceed as follows:

1. Start replication and note the timestamp.
2. Run the `copyTable` command with an end timestamp equal to the timestamp you noted in the previous step.

   - **Note:**

     Because replication starts from the current WAL, some key values may be copied to the slave cluster by both the replication and the `copyTable` job. This is is not a problem because this is an idempotent operation (one that can be applied multiple times without changing the result).

Replicating Pre-existing Data in a Master-Master Deployment

In the case of master-master replication, run the `copyTable` job before starting the replication. (If you start the job after enabling replication, the second master will re-send the data to the first master, because `copyTable` does not edit the `clusterId` in the mutation objects. Proceed as follows:

1. Run the `copyTable` job and note the start timestamp of the job.
2. Start replication.
3. Run the `copyTable` job again with a start time equal to the start time you noted in step 1.

This results in some data being pushed back and forth between the two clusters; but it minimizes the amount of data.

Verifying Replicated Data

If you are looking only at a few rows, you can verify the replicated data in the shell.

For a systematic comparison on a larger scale, use the `VerifyReplication` MapReduce job. Run it on the master cluster and provide it with the peer ID (the one you provided when establishing the replication stream), a table name, and a column family. Other options allow you to specify a time range and specific families. This job's short name is `verifyrep`; provide that name when pointing `hadoop jar` to the HBase JAR file.

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 command prints out `GOODROWS` and `BADROWS` counters; these correspond to replicated and non-replicated rows respectively.

Configuring Secure HBase Replication

If you want to make HBase Replication secure, follow the instructions under HBase Security Configuration.

Caveats

- Two variables govern replication: `hbase.replication` as described above under Deploying HBase Replication, and a replication znode. Stopping replication (using `stop_replication` as above) sets the znode to `false`. Two problems can result:
  - If you add a new Region Server to the master 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 `start_replication`), entries in the log will not be replicated.
  - Similarly, if a logs rolls on an existing Region Server on the master cluster while replication is stopped, the new log will not be replicated, because the the replication znode was set to `false` when the new log was created.
- Loop detection is not guaranteed in all cases if you use cyclic replication among more than two clusters. Follow these guidelines.
- In the case of a long-running, write-intensive workload, the slave 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 slave cluster (default is 3). Replication is now handled by separate handlers in the slave cluster to avoid the above-mentioned sluggishness. Increase it to a high value if the ratio of master to slave 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).

Configuring HBase Snapshots

About HBase Snapshots

In previous HBase releases, the only way to a backup or to clone a table was to use `CopyTable` or `ExportTable`, or to copy all the `hfiles` in HDFS after disabling the table. The disadvantages of these methods are:

- `CopyTable` and `ExportTable` can degrade region server 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 Region Servers. Exporting the table to another cluster should not have any impact on the the region servers.

Use Cases

- Recovery from user or application errors
  - Useful because it may be some time before the database administrator notices the error

    > **Note:**
    >
    > The database administrator needs to schedule the intervals at which to take and delete snapshots. Use a script or your preferred management tool for this; it is not built into HBase.

  - The database administrator may want to save a snapshot right before a major application upgrade or change.

    > **Note:**
    >
    > Snapshots are not primarily used for system upgrade protection because they would not roll back binaries, and would not necessarily be proof against bugs or errors in the system or the upgrade.

  - Sub-cases for recovery:
    - Rollback to previous snapshot and merge in reverted data
    - View previous snapshots and selectively merge them into production
- Backup
  - Capture a copy of the database and store it outside HBase for disaster recovery
  - Capture previous versions of data for compliance, regulation, archiving
  - Export from snapshot on live system provides a more consistent view of HBase than `CopyTable` and `ExportTable`
- Audit and/or report view of data at a specific time
  - Capture monthly data for compliance
  - Use for end-of-day/month/quarter reports
- Use for Application testing
  - Test schema or application changes on like production data from snapshot and then throw away
  - For example: take a snapshot; create a new table from the snapshot content (schema plus data); manipulate the new table by changing the schema, adding and removing rows, and so on (the original table, the snapshot, and the new table remain independent of each other)
- Offload work

- Capture, copy, and restore data to another site
- Export data to another cluster

Where Snapshots Are Stored

The snapshot metadata is stored in the `.hbase_snapshot` directory under the `hbase` root directory (`/hbase/.hbase-snapshot`). Each snapshot has its own directory that includes all the references to the `hfiles`, logs, and metadata needed to restore the table.

`hfiles` needed by the snapshot are in the traditional `/hbase/data/<namespace>/<tableName>/<regionName>/<familyName>/` location if the table is still using them; otherwise they will be placed in `/hbase/.archive/<namespace>/<tableName>/<regionName>/<familyName>/`

Zero-copy Restore and Clone Table

From a snapshot you can create a new table (`clone` operation) or restore the original table. These two operations do not involve data copies; instead a link is created to point to the original `hfiles`.

Changes to a cloned or restored table do not affect the snapshot or (in case of a clone) the original table.

If you want to clone a table to another cluster, you need to export the snapshot to the other cluster and then execute the `clone` operation; see Exporting a Snapshot to Another Cluster.

Reverting to a Previous HBase Version

Snapshots don't affect HBase backward compatibility if they are not used.

If you do use the snapshot capability, backward compatibility is affected as follows:

- If you only take snapshots, you can still go back to a previous HBase version
- If you have used `restore` or `clone`, you cannot go back to a previous version unless the cloned or restored tables have no links (there is no automated way to check; you would need to inspect the file system manually).

Storage Considerations

Since the `hfiles` are immutable, a snapshot consists of reference to the files that are in the table at the moment the snapshot is taken. No copies of the data are made during the snapshot operation, but copies may be made when a compaction or deletion is triggered. In this case, if a snapshot has a reference to the files to be removed, the files are moved to an archive folder, instead of being deleted. This allows the snapshot to be restored in full.

Because no copies are performed, multiple snapshots share the same `hfiles`, but in the worst case scenario, each snapshot could have different set of hfiles (tables with lots of updates, and compactions).

Configuring and Enabling Snapshots

Snapshots are on by default; to disable them, set the `hbase.snapshot.enabled` property in `hbase-site.xml` to `false`:

```
<property>
   <name>hbase.snapshot.enabled</name>
   <value>
      false
   </value>
</property>
```

To enable snapshots after you have disabled them, set `hbase.snapshot.enabled` to `true`.

Note:

If you have taken snapshots and then decide to disable snapshots, you must delete the snapshots before restarting the HBase master; the HBase master will not start if snapshots are disabled and snapshots exist.

Snapshots don't affect HBase performance if they are not used.

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 (via `disable <table>`) an offline snapshot is taken. This snapshot is driven 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 since the table must be disabled to take the snapshot. ▪ In an online snapshot, the table remains available while the snapshot is taken, and should incur minimal noticeable performance degradation of normal read/write loads. This snapshot is coordinated by the master and run on the region servers. The current implementation - simple-flush snapshots - provides no causal consistency guarantees. Despite this shortcoming, it offers the same degree of consistency as `CopyTable` and overall is a huge improvement over `CopyTable`.
Restore snapshot `snapshotX` (it will replace the source table content)	`restore_snapshot ‘snapshotX’`	Restoring a snapshot attempts to replace the current version of a table with another version of the table. To run this command, you must disable the target table. The `restore` command takes a snapshot of the table (appending a timestamp code), and then essentially clones data into the original data and removes data not in the snapshot. If the operation succeeds, the target table will be enabled. Use this capability only in an emergency; see Restrictions.
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 can serve the data kept at the time of the snapshot. The original table and the cloned table can be modified independently without interfering – new data written to one table will not show up on the other.

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's metadata, from the source cluster to the destination cluster. You can 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 196.

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.

To copy a snapshot called MySnapshot to an HBase cluster srv2 (hdfs://srv2:8082/hbase) using 16 mappers:

```
hbase class org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
 hdfs://srv2:8082/hbase -mappers 16
```

To export the snapshot and change the ownership of the files during the copy:

```
hbase class org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
 hdfs://srv2:8082/hbase -chuser MyUser -chgroup MyGroup -chmod 700 -mappers 16
```

You can also use the Java `-D` option in many tools to specify MapReduce or other configuration. properties. For example, the following command copies `MY_SNAPSHOT` to `hdfs://cluster2/hbase` using groups of 10 hfiles per mapper:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot
-Dsnapshot.export.default.map.group=10 -snapshot MY_SNAPSHOT -copy-to
hdfs://cluster2/hbase
```

(The number of mappers is calculated as `TotalNumberOfHFiles`/10.)

To export from one remote cluster to another remote cluster, specify both `-copy-from` and `-copy-to` parameters. You could then reverse the direction to restore the snapshot bacvk to the first remote cluster.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-test -copy-from
 hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup
```

To specify a different name for the snapshot on the target cluster, use the `-target` option.

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot snapshot-test -copy-from
 hdfs://machine1/hbase -copy-to hdfs://machine2/my-backup -target new-snapshot
```

Restrictions

- **Warning:**

  Do not use merge in combination with snapshots. Merging two regions can cause data loss if snapshots or cloned tables exist for this table.

  The merge is likely to corrupt the snapshot and any tables cloned from the snapshot. In addition, if the table has been restored from a snapshot, the merge may also corrupt the table. The snapshot may survive intact if the regions being merged are not in the snapshot, and clones may survive if they do not share files with the original table or snapshot. You can use the `Snapinfo` tool (see Information and Debugging on page 200) to check the status of the snapshot. If the status is `BROKEN`, the snapshot is unusable.

- All the Masters and Region Servers must be running CDH 5.
- If you have enabled the `AccessController Coprocessor` for HBase, only a global administrator can take, clone, or restore a snapshot, and these actions do not capture the ACL rights. This means that restoring a table preserves the ACL rights of the existing table, while cloning a table creates a new table that has no ACL rights until the administrator adds them.
- Do not take, clone, or restore a snapshot during a rolling restart. Snapshots rely on the Region Servers being up; otherwise the snapshot will fail.

> **Note:** This restriction also applies to rolling upgrade, which can currently be done only via Cloudera Manager.

If you are using HBase Replication and you need to restore a snapshot: If you are using HBase Replication the replicas will be out of synch when you restore a snapshot. Do this only in an emergency.

> **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 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 slave clusters
3. Restore the snapshot on the master cluster
4. Create the table on the slave cluster and use `CopyTable` to initialize it.

> **Note:**
>
> If this is not an emergency (for example, if you know that you have lost just a set of rows such as the rows starting with "xyz"), you can create a clone from the snapshot and create a MapReduce job to copy the data that you've lost.
>
> In this case you don't need to stop replication or disable your main table.

Snapshot Failures

Region moves, splits, and other metadata actions that happen while a snapshot is in progress will probably cause the snapshot to fail; the software detects and rejects corrupted snapshot attempts.

Information and Debugging

You can use the `SnapshotInfo` tool to get information about a snapshot, including status, files, disk usage, and debugging information.

Examples:

Use the `-h` option to print usage instructions for the `SnapshotInfo` utility.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -h
Usage: bin/hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo [options]
 where [options] are:
  -h|-help                Show this help and exit.
  -remote-dir             Root directory that contains the snapshots.
  -list-snapshots         List all the available snapshots and exit.
  -snapshot NAME          Snapshot to examine.
  -files                  Files and logs list.
  -stats                  Files and logs stats.
  -schema                 Describe the snapshotted table.
```

Use the `-list-snapshots` option to list all snapshots and exit. This option is new in CDH 5.1.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -list-snapshots
SNAPSHOT                | CREATION TIME         | TABLE NAME
snapshot-test           |  2014-06-24T19:02:54 | test
```

Use the `-remote-dir` option with the `-list-snapshots` option to list snapshots located on a remote system.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -remote-dir
s3n://mybucket/mysnapshot-dir -list-snapshots
SNAPSHOT |   CREATION TIME   | TABLE NAME
snapshot-test        2014-05-01 10:30    myTable
```

Use the `-snapshot` option to print information about a specific snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot
Snapshot Info
----------------------------------------
   Name: test-snapshot
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21
**************************************************************
```

Use the `-snapshot` with the `-stats` options to display additional statistics about a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -stats -snapshot snapshot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54


1 HFiles (0 in archive), total size 1.0k (100.00% 1.0k shared with the source table)
```

Use the `-schema` option with the `-snapshot` option to display the schema of a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo  -schema -snapshot snapshot-test
Snapshot Info
----------------------------------------
   Name: snapshot-test
   Type: FLUSH
  Table: test
 Format: 0
Created: 2014-06-24T19:02:54

Table Descriptor
----------------------------------------
'test', {NAME => 'cf', DATA_BLOCK_ENCODING => 'FAST_DIFF', BLOOMFILTER => 'ROW',
REPLICATION_SCOPE => '0',
COMPRESSION => 'GZ', VERSIONS => '1', TTL => 'FOREVER', MIN_VERSIONS => '0',
KEEP_DELETED_CELLS => 'false',
BLOCKSIZE => '65536', IN_MEMORY => 'false', BLOCKCACHE => 'true'}
```

Use the `-files` option with the `-snapshot` option to list information about files contained in a snapshot.

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot -files
Snapshot Info
----------------------------------------
   Name: test-snapshot
   Type: DISABLED
  Table: test-table
Version: 0
Created: 2012-12-30T11:21:21
```

```

Snapshot Files
----------------------------------------
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/bdf29c39da2a4f2b81889eb4f7b18107
 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/1e06029d0a2a4a709051b417aec88291
 (archive)
   86.8k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/506f601e14dc4c74a058be5843b99577
 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/5c7f6916ab724eacbcea218a713941c4
 (archive)
  293.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/aec5e33a6564441d9bd423e31fc93abb
 (archive)
   52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/97782b2fbf0743edaacd8fef06ba51e4
 (archive)

6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the source table)
0 Logs, total size 0.0
```

Installing and Using HCatalog

As of CDH 5, HCatalog is part of Apache Hive.

HCatalog provides table data access for CDH components such as Pig, Sqoop, and MapReduce. Table definitions are maintained in the Hive metastore, which HCatalog requires. HCatalog makes the same table information available to Hive, Pig, MapReduce, and REST clients. This page explains how to install and configure HCatalog for REST access and for MapReduce and Pig access. For Sqoop, see the section on Sqoop-HCatalog integration in the Sqoop User Guide.

Use the sections that follow to install, configure and use HCatalog:

- Prerequisites
- Installing and Upgrading the HCatalog RPM or Debian Packages on page 203
- Host Configuration Changes
- Starting and Stopping the WebHCat REST Server
- Accessing Table Data with the Command-line API
- Accessing Table Data with MapReduce
- Accessing Table Data with Pig
- Accessing Table Data with REST
- Apache HCatalog Documentation

For more information, see the HCatalog documentation.

HCatalog Prerequisites

- An operating system supported by CDH 5
- Oracle JDK
- The Hive metastore and its database. The Hive metastore must be running in remote mode (as a service).

Installing and Upgrading the HCatalog RPM or Debian Packages

Installing the HCatalog RPM or Debian packages is more convenient than installing the HCatalog tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations

HCatalog comprises the following packages:

- `hive-hcatalog` - HCatalog wrapper for accessing the Hive metastore, libraries for MapReduce and Pig, and a command-line program
- `hive-webhcat` - A REST API server for HCatalog
- `hive-webhcat-server` - Installs `hive-webhcat` and a server `init` script

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install HCatalog. For instructions, see CDH 5 Installation on page 27.

Upgrading HCatalog from CDH 4 to CDH 5

To upgrade HCatalog from CDH 4 to CDH 5, proceed as follows.

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5 on page 57, you can skip Step 1 below and proceed with installing the new CDH 5 version of HCatalog.

Step 1: Remove the CDH 4 version of HCatalog

To remove HCatalog on a Red Hat-compatible system:

```
$ sudo yum remove webhcat-server hcatalog
```

To remove HCatalog on an Ubuntu or other Debian system:

```
$ sudo apt-get remove webhcat-server hcatalog
```

To remove HCatalog on a SLES system:

```
$ sudo zypper remove webhcat-server hcatalog
```

Step 2: Install the new version of WebHCat and HCatalog

Follow instructions under Installing the WebHCat REST Server on page 205 and Installing HCatalog for Use with Pig and MapReduce on page 205.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

The upgrade is now complete.

Upgrading HCatalog from an Earlier CDH 5 Release

Important:

If you have installed the `hive-hcatalog-server` package in the past, you must remove it before you proceed; otherwise the upgrade will fail.

Follow instructions under Installing the WebHCat REST Server on page 205 and Installing HCatalog for Use with Pig and MapReduce on page 205.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

The upgrade is now complete.

Installing the WebHCat REST Server

Note:

It is not necessary to install WebHCat if you will not be using the REST API. Pig and MapReduce do not need it.

To install the WebHCat REST server on a Red Hat system:

```
$ sudo yum install hive-webhcat-server
```

To install the WebHCat REST server components on an Ubuntu or other Debian system:

```
$ sudo apt-get install hive-webhcat-server
```

To install the WebHCat REST server components on a SLES system:

```
$ sudo zypper install hive-webhcat-server
```

Note:

- You can change the default port 50111 by creating or editing the following file and restarting WebHCat:

  ```
  /etc/webhcat/conf/webhcat-site.xml
  ```

  The property to change is:

  ```
  <configuration>
    <property>
      <name>templeton.port</name>
      <value>50111</value>
      <description>The HTTP port for the main server.</description>
    </property>
  </configuration>
  ```

- To uninstall WebHCat you must remove two packages: `hive-webhcat-server` and `hive-webhcat`.

Installing HCatalog for Use with Pig and MapReduce

On hosts that will be used to launch Pig scripts or MapReduce applications using table information, install HCatalog as follows:

To install the HCatalog client components on a Red Hat system:

```
$ sudo yum install hive-hcatalog
```

To install the HCatalog client components on an Ubuntu or other Debian system:

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

To install the HCatalog client components on a SLES system:

```
$ sudo zypper install hive-hcatalog
```

Configuration Change on Hosts Used with HCatalog

You must update `/etc/hive/conf/hive-site.xml` on all hosts where WebHCat will run, as well as all hosts where Pig or MapReduce will be used with HCatalog, so that Metastore clients know where to find the Metastore.

Add or edit the `hive.metastore.uris` property as follows:

```
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<hostname>:9083</value>
</property>
```

where <hostname> is the host where the HCatalog server components are running, for example `hive.examples.com`.

Starting and Stopping the WebHCat REST server

```
$ sudo service webhcat-server start
$ sudo service webhcat-server stop
```

Accessing Table Information with the HCatalog Command-line API

```
# Create a table
$ hcat -e "create table groups(name string,placeholder string,id int) row format
delimited fields terminated by ':' stored as textfile"
OK

# Get the schema for a table
$ hcat -e "desc groups"
OK
name string
placeholder string
id int

# Create another table
$ hcat -e "create table groupids(name string,id int)"
OK
```

See the HCatalog documentation for information on using the HCatalog command-line application.

Accessing Table Data with MapReduce

You can download an example of a MapReduce program that reads from the groups table (consisting of data from `/etc/group`), extracts the first and third columns, and inserts them into the `groupids` table. Proceed as follows.

1. Download the program from https://github.com/cloudera/hcatalog-examples.git.
2. Build the example JAR file:

```
$ cd hcatalog-examples
$ mvn package
```

3. Load data from the local file system into the groups table:

```
$ hive -e "load data local inpath '/etc/group' overwrite into table groups"
```

4. Set up the environment that is needed for copying the required JAR files to HDFS, for example:

```
$ export HCAT_HOME=/usr/lib/hive-hcatalog
$ export HIVE_HOME=/usr/lib/hive
$ HIVE_VERSION=0.11.0-cdh5.0.0
$ HCATJAR=$HCAT_HOME/share/hcatalog/hcatalog-core-$HIVE_VERSION.jar
$ HCATPIGJAR=$HCAT_HOME/share/hcatalog/hcatalog-pig-adapter-$HIVE_VERSION.jar
$ export
HADOOP_CLASSPATH=$HCATJAR:$HCATPIGJAR:$HIVE_HOME/lib/hive-exec-$HIVE_VERSION.jar\
:$HIVE_HOME/lib/hive-metastore-$HIVE_VERSION.jar:$HIVE_HOME/lib/jdo-api-*.jar:$HIVE_HOME/lib/libfb303-*.jar\
:$HIVE_HOME/lib/libthrift-*.jar:$HIVE_HOME/lib/slf4j-api-*.jar:$HIVE_HOME/conf:/etc/hadoop/conf
$ LIBJARS=`echo $HADOOP_CLASSPATH | sed -e 's/:/,/g'`
$ export LIBJARS=$LIBJARS,$HIVE_HOME/lib/antlr-runtime-*.jar
```

Note: You can find current version numbers for CDH dependencies in CDH's root `pom.xml` file for the current release, for example cdh-root-5.0.0.pom.)

5. Run the job:

```
$ hadoop jar target/UseHCat-1.0.jar com.cloudera.test.UseHCat -files $HCATJAR
-libjars $LIBJARS groups groupids
```

Accessing Table Data with Pig

When using table information from the Hive metastore with Pig, add the -useHCatalog option when invoking pig:

```
$ pig -useHCatalog test.pig
```

In the script, use HCatLoader to have table schema retrieved automatically:

```
A = LOAD 'groups' USING org.apache.hcatalog.pig.HCatLoader();
DESCRIBE A;
```

Output:

```
A: {name: chararray,placeholder: chararray,id: int}
```

Accessing Table Information with REST

Table information can be retrieved from any host that has HTTP access to the host where the WebHCat server is running. A Web browser or an HTTP client such as curl or wget can be used to verify the functionality.

The base URL for REST access to table information is `http://<SERVERHOST>:50111/templeton/v1/ddl`.

Examples of specific URLs:

```
http://<SERVERHOST>:50111/templeton/v1/ddl/database/?user.name=hive
http://<SERVERHOST>:50111/templeton/v1/ddl/database/default/table/?user.name=hive
http://<SERVERHOST>:50111/templeton/v1/ddl/database/default/table/groups?user.name=hive
```

Example output:

```
{"columns":[{"name":"name","type":"string"},{"name":"placeholder","type":"string"},{"name":"id","type":"int"}],"database":"default","table":"grouptable"}
```

Supported REST Endpoints

The General and DDL endpoints are supported, for accessing Hive metadata. If you need submission capabilities for MapReduce, Hive, or Pig jobs, consider using Oozie, which is a more mature interface. See Installing Oozie on page 300.

Category	Resource Type	Description
General	:version (GET)	Return a list of supported response types.
	status (GET)	Return the WebHCat server status.
	version (GET)	Return a list of supported versions and the current version.
	version/hive (GET)	Return the Hive version being run.
	version/hadoop (GET)	Return the Hadoop version being run.
DDL	ddl (POST)	Perform an HCatalog DDL command.
	ddl/database (GET)	List HCatalog databases.
	ddl/database/:db (GET)	Describe an HCatalog database.
	ddl/database/:db (PUT)	Create an HCatalog database.
	ddl/database/:db (DELETE)	Delete (drop) an HCatalog database.
	ddl/database/:db/table (GET)	List the tables in an HCatalog database.
	ddl/database/:db/table/:table (GET)	Describe an HCatalog table.
	ddl/database/:db/table/:table (PUT)	Create a new HCatalog table.
	ddl/database/:db/table/:table (POST)	Rename an HCatalog table.
	ddl/database/:db/table/:table (DELETE)	Delete (drop) an HCatalog table.
	ddl/database/:db/table/:existingtable/like/:newtable (PUT)	Create a new HCatalog table like an existing one.

Category	Resource Type	Description
	ddl/database/:db/table/:table/partition (GET)	List all partitions in an HCatalog table.
	ddl/database/:db/table/:table/partition/:partition (GET)	Describe a single partition in an HCatalog table.
	ddl/database/:db/table/:table/partition/:partition (PUT)	Create a partition in an HCatalog table.
	ddl/database/:db/table/:table/partition/:partition (DELETE)	Delete (drop) a partition in an HCatalog table.
	ddl/database/:db/table/:table/column (GET)	List the columns in an HCatalog table.
	ddl/database/:db/table/:table/column/:column (GET)	Describe a single column in an HCatalog table.
	ddl/database/:db/table/:table/column/:column (PUT)	Create a column in an HCatalog table.
	ddl/database/:db/table/:table/property (GET)	List table properties.
	ddl/database/:db/table/:table/property/:property (GET)	Return the value of a single table property.
	ddl/database/:db/table/:table/property/:property (PUT)	Set a table property.

Viewing the HCatalog Documentation

See Apache wiki page.

Impala Installation

Cloudera Impala is an open-source add-on to the Cloudera Enterprise Core that returns rapid responses to queries.

Impala is installed separately from other components as an add-on to your environment. Impala is made up of a set of components which can be installed on multiple nodes throughout your cluster.

The Impala package installs these binaries:

- `impalad` - The Impala daemon. Plans and executes queries against HDFS and HBase data. Run one daemon process on each node in the cluster that has a data node.
- `statestored` - Name service that tracks location and status of all `impalad` instances in the cluster. Run one instance of this daemon on a node in your cluster.
- `catalogd` - Metadata coordination service that broadcasts changes from Impala DDL and DML statements to all affected Impala nodes, so that new tables, newly loaded data, and so on are immediately visible to queries submitted through any Impala node. (Prior to Impala 1.2, you had to run the `REFRESH` or `INVALIDATE METADATA` statement on each node to synchronize changed metadata. Now those statements are only required if you perform the DDL or DML through Hive.) Run one instance of this daemon on a node in your cluster, preferable on the same host as the `statestored` daemon.
- `impala-shell` - Command-line interface for issuing queries to the Impala daemon. You install this on one or more hosts anywhere on your network, not necessarily data nodes. It can connect remotely to any instance of the Impala daemon.

Before doing the installation, ensure that you have all necessary prerequisites. See Cloudera Impala Requirements on page 211 for details.

You can install Impala in one of two ways:

- Using the Cloudera Manager installer, as described in Installing Impala with Cloudera Manager on page 215. This is the recommended technique for doing a reliable and verified Impala installation. Cloudera Manager 4.8 or later can automatically install, configure, manage, and monitor Impala 1.2.1 and higher.
- Using the manual process described in Installing Impala without Cloudera Manager on page 216. You must do additional verification steps in this case, to check that Impala can interact with other Hadoop components correctly, and that your cluster is configured for efficient Impala execution.

Cloudera Impala Requirements

To perform as expected, Impala depends on the availability of the software, hardware, and configurations described in the following sections.

Product Compatibility Matrix

The ultimate source of truth about compatibility between various versions of CDH, Cloudera Manager, and various CDH components is the online Product Compatibility Matrix. For Impala, see the Impala compatibility matrix page.

Supported Operating Systems

Supported 64-bit operating systems:

- Red Hat Enterprise Linux (RHEL), Oracle Linux, or Centos.

On version 5 of Red Hat Enterprise Linux and comparable distributions, some additional setup is needed for the `impala-shell` interpreter to connect to a Kerberos-enabled Impala cluster:

```
sudo yum install python-devel openssl-devel python-pip
sudo pip-python install ssl
```

- SLES.
- Ubuntu or Debian.
- For the relevant supported OS versions see the *Supported Operating Systems* page for CDH 4 or CDH 5.

Supported CDH Versions

Cloudera supports Impala only under the CDH Hadoop distribution. The following list shows the mapping between Impala versions and the earliest corresponding CDH version:

- CDH 5.1.3 for Impala 1.4.2. (Not available for CDH 4.)
- CDH 5.1.2 for Impala 1.4.1. (Not available for CDH 4.)
- CDH 4.1 or later 4.x, or CDH 5.1.0 or later 5.1.x, for Impala 1.4.0.
- CDH 5.0.4 for Impala 1.3.2. (This Impala update consists of a single bug fix that is not applicable for CDH 4.)
- CDH 4.1 or later 4.x, or CDH 5.0.1, for Impala 1.3.1.
- CDH 5.0 for Impala 1.3.0. (The first Impala 1.3.x release for use with CDH 4 is 1.3.1.)
- CDH 4.1 or later 4.x for Impala 1.2.4.
- CDH 4.1 or later 4.x, or CDH 5 beta 2, for Impala 1.2.3.
- CDH 4.1 or later 4.x for Impala 1.2.2 and 1.2.1. (Impala 1.2.2 and 1.2.1 do not work with CDH 5 beta due to differences with packaging and dependencies.)
- CDH 5 beta 1 for Impala 1.2.0.
- CDH 4.1 or later for Impala 1.1.
- CDH 4.1 or later for Impala 1.0.
- CDH 4.1 or later for Impala 0.7.
- CDH 4.2 or later for Impala 0.6.
- CDH 4.1 for Impala 0.5 and earlier. This combination is only supported on RHEL/CentOS.

Hive Metastore and Related Configuration

Impala can interoperate with data stored in Hive, and uses the same infrastructure as Hive for tracking metadata about schema objects such as tables and columns. The following components are prerequisites for Impala:

- MySQL or PostgreSQL, to act as a metastore database for both Impala and Hive.

Note:

Installing and configuring a Hive metastore is an Impala requirement. Impala does not work without the metastore database. For the process of installing and configuring the metastore, see Impala Installation on page 211.

Always configure a **Hive metastore service** rather than connecting directly to the metastore database. The metastore service is required to interoperate between possibly different levels of metastore APIs used by CDH and Impala, and avoids known issues with connecting directly to the metastore database. The metastore service is set up for you by default if you install through Cloudera Manager 4.5 or later.

A summary of the metastore installation process is as follows:

- Install a MySQL or PostgreSQL database. Start the database if it is not started after installation.
- Download the MySQL connector or the PostgreSQL connector and place it in the `/usr/share/java/` directory.
- Use the appropriate command line tool for your database to create the metastore database.
- Use the appropriate command line tool for your database to grant privileges for the metastore database to the `hive` user.
- Modify `hive-site.xml` to include information matching your particular database: its URL, user name, and password. You will copy the `hive-site.xml` file to the Impala Configuration Directory later in the Impala installation process.

- **Optional:** Hive. Although only the Hive metastore database is required for Impala to function, you might install Hive on some client machines to create and load data into tables that use certain file formats. See How Impala Works with Hadoop File Formats for details. Hive does not need to be installed on the same data nodes as Impala; it just needs access to the same metastore database.

Java Dependencies

Although Impala is primarily written in C++, it does use Java to communicate with various Hadoop components:

- The officially supported JVM for Impala is the Oracle JVM. Other JVMs might cause issues, typically resulting in a failure at `impalad` startup. In particular, the JamVM used by default on certain levels of Ubuntu systems can cause `impalad` to fail to start.
- Internally, the `impalad` daemon relies on the `JAVA_HOME` environment variable to locate the system Java libraries. Make sure the `impalad` service is not run from an environment with an incorrect setting for this variable.
- All Java dependencies are packaged in the `impala-dependencies.jar` file, which is located at `/usr/lib/impala/lib/`. These map to everything that is built under `fe/target/dependency`.

Packages and Repositories

Packages or properly configured repositories. You can install Impala manually using packages, through the Cloudera Impala public repositories, or from your own custom repository. To install using the Cloudera Impala repository, download and install the file to each machine on which you intend to install Impala or Impala Shell. Install the appropriate package or list file as follows:

- Red Hat 5 repo file (http://archive.cloudera.com/impala/redhat/5/x86_64/impala/cloudera-impala.repo) in `/etc/yum.repos.d/`.
- Red Hat 6 repo file (http://archive.cloudera.com/impala/redhat/6/x86_64/impala/cloudera-impala.repo) in `/etc/yum.repos.d/`.
- SUSE repo file (http://archive.cloudera.com/impala/sles/11/x86_64/impala/cloudera-impala.repo) in `/etc/zypp/repos.d/`.
- Ubuntu 10.04 list file (http://archive.cloudera.com/impala/ubuntu/lucid/amd64/impala/cloudera.list) in `/etc/apt/sources.list.d/`.

- Ubuntu 12.04 list file (http://archive.cloudera.com/impala/ubuntu/precise/amd64/impala/cloudera.list) in `/etc/apt/sources.list.d/`.
- Debian list file (http://archive.cloudera.com/impala/debian/squeeze/amd64/impala/cloudera.list) in `/etc/apt/sources.list.d/`.

For example, on a Red Hat 6 system, you might run a sequence of commands like the following:

```
$ cd /etc/yum.repos.d
$ sudo wget
http://archive.cloudera.com/impala/redhat/6/x86_64/impala/cloudera-impala.repo
$ ls
CentOS-Base.repo  CentOS-Debuginfo.repo  CentOS-Media.repo  Cloudera-cdh.repo
cloudera-impala.repo
```

Note:

You can retrieve files from the `archive.cloudera.com` site through `wget` or `curl`, but not through `rsync`.

Optionally, you can install and manage Impala through the Cloudera Manager product. Impala 1.4.0 requires Cloudera Manager 4.8 or higher, although Cloudera Manager 5.1 or higher is recommended to take advantage of configuration settings for new Impala features such as admission control, load-balancing support on Kerberos clusters, and the **Impala Best Practices** page in Cloudera Manager.

Note:

In a Cloudera Manager environment, the catalog service is not recognized or managed by Cloudera Manager versions prior to 4.8. Cloudera Manager 4.8 and higher require the catalog service to be present for Impala. Therefore, if you upgrade to Cloudera Manager 4.8 or higher, you must also upgrade Impala to 1.2.1 or higher. Likewise, if you upgrade Impala to 1.2.1 or higher, you must also upgrade Cloudera Manager to 4.8 or higher.

When you install through Cloudera Manager, you can install either with the OS-specific "package" mechanism, or the Cloudera Manager "parcels" mechanism, which simplifies upgrades and deployment across an entire cluster.

Networking Configuration Requirements

As part of ensuring best performance, Impala attempts to complete tasks on local data, as opposed to using network connections to work with remote data. To support this goal, Impala matches the **hostname** provided to each Impala daemon with the **IP address** of each datanode by resolving the hostname flag to an IP address. For Impala to work with local data, use a single IP interface for the datanode and the Impala daemon on each machine. Ensure that the Impala daemon's hostname flag resolves to the IP address of the datanode. For single-homed machines, this is usually automatic, but for multi-homed machines, ensure that the Impala daemon's hostname resolves to the correct interface. Impala tries to to detect the correct hostname at start-up, and prints the derived hostname at the start of the log in a message of the form:

```
Using hostname: impala-daemon-1.cloudera.com
```

In the majority of cases, this automatic detection works correctly. If you need to explicitly set the hostname, do so by setting the `--hostname` flag.

Hardware Requirements

During join operations, portions of data from each joined table are loaded into memory. Data sets can be very large, so ensure your hardware has sufficient memory to accommodate the joins you anticipate completing.

While requirements vary according to data set size, the following is generally recommended:

- CPU - Impala uses the SSE4.2 instruction set, which is included in newer processors. Impala can use older processors as long as they support the SSE3 instruction set, but for best performance use:
  - Intel - Nehalem (released 2008) or later processors.
  - AMD - Bulldozer (released 2011) or later processors.
- Memory - 128 GB or more recommended, ideally 256 GB or more. If the intermediate results during query processing on a particular node exceed the amount of memory available to Impala on that node, the query is cancelled. Note that because the work is parallelized, and intermediate results for aggregate queries are typically smaller than the original data, Impala can query and join tables that are much larger than the memory available on an individual node.
- Storage - DataNodes with 12 or more disks each. I/O speeds are often the limiting factor for disk performance with Impala. Ensure that you have sufficient disk space to store the data Impala will be querying.

User Account Requirements

Impala creates and uses a user and group named `impala`. Do not delete this account or group and do not modify the account's or group's permissions and rights. Ensure no existing systems obstruct the functioning of these accounts and groups. For example, if you have scripts that delete user accounts not in a white-list, add these accounts to the list of permitted accounts.

For the resource management feature to work (in combination with CDH 5 and the YARN and Llama components), the `impala` user must be a member of the `hdfs` group. This setup is performed automatically during a new install, but not when upgrading from earlier Impala releases to Impala 1.2. If you are upgrading a node to CDH 5 that already had Impala 1.1 or 1.0 installed, manually add the `impala` user to the `hdfs` group. For Llama installation instructions, see Llama installation.

For correct file deletion during `DROP TABLE` operations, Impala must be able to move files to the HDFS trashcan. You might need to create an HDFS directory `/user/impala`, writeable by the `impala` user, so that the trashcan can be created. Otherwise, data files might remain behind after a `DROP TABLE` statement.

Impala should not run as root. Best Impala performance is achieved using direct reads, but root is not permitted to use direct reads. Therefore, running Impala as root negatively affects performance.

By default, any user can connect to Impala and access all the associated databases and tables. You can enable authorization and authentication based on the Linux OS user who connects to the Impala server, and the associated groups for that user. Impala Security Configuration for details. These security features do not change the underlying file permission requirements; the `impala` user still needs to be able to access the data files.

Installing Impala with Cloudera Manager

Before installing Impala through the Cloudera Manager interface, make sure all applicable nodes have the appropriate hardware configuration and levels of operating system and CDH. See Cloudera Impala Requirements on page 211 for details.

Note:

To install the latest Impala under CDH 4, upgrade Cloudera Manager to 4.8 or higher. Cloudera Manager 4.8 is the first release that can manage the Impala catalog service introduced in Impala 1.2. Cloudera Manager 4.8 requires this service to be present, so if you upgrade to Cloudera Manager 4.8, also upgrade Impala to the most recent version at the same time.

For information on installing Impala in a Cloudera Manager-managed environment, see the *Cloudera Manager Installation Guide* for Cloudera Manager 5 or Cloudera Manager 4.

Managing your Impala installation through Cloudera Manager has a number of advantages. For example, when you make configuration changes to CDH components using Cloudera Manager, it automatically applies changes to the copies of configuration files, such as `hive-site.xml`, that Impala keeps under `/etc/impala/conf`. It also sets up the Hive metastore service that is required for Impala running under CDH 4.1.

In some cases, depending on the level of Impala, CDH, and Cloudera Manager, you might need to add particular component configuration details in some of the free-form option 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**.

Installing Impala without Cloudera Manager

Before installing Impala manually, make sure all applicable nodes have the appropriate hardware configuration, levels of operating system and CDH, and any other software prerequisites. See Cloudera Impala Requirements on page 211 for details.

You can install Impala across many hosts or on one host:

- Installing Impala across multiple machines creates a distributed configuration. For best performance, install Impala on **all** DataNodes.
- Installing Impala on a single machine produces a pseudo-distributed cluster.

To install Impala on a host:

1. Install CDH as described in the Installation section of the CDH 4 Installation Guide or the CDH 5 Installation Guide.
2. Install the Hive metastore somewhere in your cluster, as described in the Hive Installation topic in the CDH 4 Installation Guide or the CDH 5 Installation Guide. As part of this process, you configure the Hive metastore to use an external database as a metastore. Impala uses this same database for its own table metadata. You can choose either a MySQL or PostgreSQL database as the metastore. The process for configuring each type of database is described in the CDH Installation Guide).

   Cloudera recommends setting up a Hive metastore service rather than connecting directly to the metastore database; this configuration is required when running Impala under CDH 4.1. Make sure the `/etc/impala/hive-site.xml` file contains the following setting, substituting the appropriate host name for *metastore_server_host*:

   ```
   <property>
   <name>hive.metastore.uris</name>
   <value>thrift://metastore_server_host:9083</value>
   </property>
   <property>
   <name>hive.metastore.client.socket.timeout</name>
   <value>3600</value>
   <description>MetaStore Client socket timeout in seconds</description>
   </property>
   ```

3. (Optional) If you installed the full Hive component on any host, you can verify that the metastore is configured properly by starting the Hive console and querying for the list of available tables. Once you confirm that the console starts, exit the console to continue the installation:

   ```
   $ hive
   Hive history file=/tmp/root/hive_job_log_root_201207272011_678722950.txt
   hive> show tables;
   table1
   table2
   hive> quit;
   $
   ```

4. Confirm that your package management command is aware of the Impala repository settings, as described in Cloudera Impala Requirements on page 211. (For CDH 4, this is a different repository than for CDH.) You might need to download a repo or list file into a system directory underneath `/etc`.
5. Use **one** of the following sets of commands to install the Impala package:

For RHEL, Oracle Linux, or CentOS systems:

```
$ sudo yum install impala             # Binaries for daemons
$ sudo yum install impala-server      # Service start/stop script
$ sudo yum install impala-state-store # Service start/stop script
$ sudo yum install impala-catalog     # Service start/stop script
```

For SUSE systems:

```
$ sudo zypper install impala             # Binaries for daemons
$ sudo zypper install impala-server      # Service start/stop script
$ sudo zypper install impala-state-store # Service start/stop script
$ sudo zypper install impala-catalog     # Service start/stop script
```

For Debian or Ubuntu systems:

```
$ sudo apt-get install impala             # Binaries for daemons
$ sudo apt-get install impala-server      # Service start/stop script
$ sudo apt-get install impala-state-store # Service start/stop script
$ sudo apt-get install impala-catalog     # Service start/stop script
```

Note: Cloudera recommends that you not install Impala on any HDFS NameNode. Installing Impala on NameNodes provides no additional data locality, and executing queries with such a configuration might cause memory contention and negatively impact the HDFS NameNode.

6. Copy the client `hive-site.xml`, `core-site.xml`, `hdfs-site.xml`, and `hbase-site.xml` configuration files to the Impala configuration directory, which defaults to `/etc/impala/conf`. Create this directory if it does not already exist.
7. Use **one** of the following commands to install `impala-shell` on the machines from which you want to issue queries. You can install `impala-shell` on any supported machine that can connect to DataNodes that are running `impalad`.

   For RHEL/CentOS systems:

   ```
   $ sudo yum install impala-shell
   ```

   For SUSE systems:

   ```
   $ sudo zypper install impala-shell
   ```

   For Debian/Ubuntu systems:

   ```
   $ sudo apt-get install impala-shell
   ```

8. Complete any required or recommended configuration, as described in Post-Installation Configuration for Impala on page 221. Some of these configuration changes are mandatory. (They are applied automatically when you install using Cloudera Manager.)

Once installation and configuration are complete, see Starting Impala on page 227 for how to activate the software on the appropriate nodes in your cluster.

If this is your first time setting up and using Impala in this cluster, run through some of the exercises in Impala Tutorial to verify that you can do basic operations such as creating tables and querying them.

Upgrading Impala

Upgrading Cloudera Impala involves stopping Impala services, using your operating system's package management tool to upgrade Impala to the latest version, and then restarting Impala services.

Note:

- Each version of CDH 5 has an associated version of Impala, When you upgrade from CDH 4 to CDH 5, you get whichever version of Impala comes with the associated level of CDH. Depending on the version of Impala you were running on CDH 4, this could install a lower level of Impala on CDH 5. For example, if you upgrade to CDH 5.0 from CDH 4 plus Impala 1.4, the CDH 5.0 installation comes with Impala 1.3. Always check the associated level of Impala before upgrading to a specific version of CDH 5. Where practical, upgrade from CDH 4 to the latest CDH 5, which also has the latest Impala.
- If you upgrade to Impala 1.4.0, upgrade Cloudera Manager as well:
  - Users running Impala on CDH 5 must upgrade to Cloudera Manager 5.0.0 or higher.
  - Users running Impala on CDH 4 must upgrade to Cloudera Manager 4.8 or higher. Cloudera Manager 4.8 includes management support for the Impala catalog service introduced in Impala 1.2, and is the minimum Cloudera Manager version you can use.
  - Cloudera Manager is continually updated with configuration settings for features introduced in the latest Impala releases.
- If you are upgrading from CDH 5 beta to CDH 5.0 production, make sure you are using the appropriate CDH 5 repositories shown on the CDH version and packaging page, then follow the procedures throughout the rest of this section.
- Every time you upgrade to a new major or minor Impala release, see Cloudera Impala Incompatible Changes in the *Release Notes* for any changes needed in your source code, startup scripts, and so on.
- Also check Cloudera Impala Known Issues and Workarounds in the *Release Notes* for any issues or limitations that require workarounds.
- Due to a change to the implementation of logging in Impala 1.1.1 and higher, currently you should change the default setting for the `logbuflevel` property for the Impala service after installing through Cloudera Manager. In Cloudera Manager, go to the log settings page for the Impala service: **Services** > **Impala** > **Configuration** > **View and Edit** > **Impala Daemon (Default)** > **Logs**. Change the setting **Impala Daemon Log Buffer Level (logbuflevel)** from -1 to 0. You might change this setting to a value higher than 0, if you prefer to reduce the I/O overhead for logging, at the expense of possibly losing some lower-priority log messages in the event of a crash.
- For the resource management feature to work (in combination with CDH 5 and the YARN and Llama components), the `impala` user must be a member of the `hdfs` group. This setup is performed automatically during a new install, but not when upgrading from earlier Impala releases to Impala 1.2. If you are upgrading a node to CDH 5 that already had Impala 1.1 or 1.0 installed, manually add the `impala` user to the `hdfs` group. For Llama installation instructions, see Llama installation.

Upgrading Impala through Cloudera Manager - Parcels

Parcels are an alternative binary distribution format available in Cloudera Manager 4.5 and higher. To upgrade Impala in a Cloudera Managed environment, using parcels:

1. If you originally installed using packages and now are switching to parcels, remove all the Impala-related packages first. You can check which packages are installed using one of the following commands, depending on your operating system:

```
rpm -qa                # RHEL, Oracle Linux, CentOS, Debian
dpkg --get-selections # Debian
```

and then remove the packages using one of the following commands:

```
sudo yum remove pkg_names    # RHEL, Oracle Linux, CentOS
sudo zypper remove pkg_names # SLES
sudo apt-get purge pkg_names # Ubuntu, Debian
```

2. Connect to the Cloudera Manager Admin Console.
3. Go to the **Hosts** > **Parcels** tab. You should see a parcel with a newer version of Impala that you can upgrade to.
4. Click **Download**, then **Distribute**. (The button changes as each step completes.)
5. Click **Activate**.
6. When prompted, click **Restart** to restart the Impala service.

Upgrading Impala through Cloudera Manager - Packages

To upgrade Impala in a Cloudera Managed environment, using packages:

1. Connect to the Cloudera Manager Admin Console.
2. In the **Services** tab, click the **Impala** service.
3. Click **Actions** and click **Stop**.
4. Use **one** of the following sets of commands to update Impala on each Impala node in your cluster:

   For RHEL, Oracle Linux, or CentOS systems:

   ```
   $ sudo yum update impala
   $ sudo yum update hadoop-lzo-cdh4 # Optional; if this package is already installed.
   ```

   For SUSE systems:

   ```
   $ sudo zypper update impala
   $ sudo zypper update hadoop-lzo-cdh4 # Optional; if this package is already
   installed
   ```

   For Debian or Ubuntu systems:

   ```
   $ sudo apt-get install impala
   $ sudo apt-get install hadoop-lzo-cdh4 # Optional; if this package is already
   installed
   ```

5. Use **one** of the following sets of commands to update Impala shell on each node on which it is installed:

   For RHEL, Oracle Linux, or CentOS systems:

   ```
   $ sudo yum update impala-shell
   ```

   For SUSE systems:

   ```
   $ sudo zypper update impala-shell
   ```

   For Debian or Ubuntu systems:

   ```
   $ sudo apt-get install impala-shell
   ```

6. Connect to the Cloudera Manager Admin Console.
7. In the **Services** tab, click the Impala service.
8. Click **Actions** and click **Start**.

Upgrading Impala without Cloudera Manager

To upgrade Impala on a cluster not managed by Cloudera Manager, run these Linux commands on the appropriate hosts in your cluster:

1. Stop Impala services.
   a. Stop `impalad` on each Impala node in your cluster:
   ```
   $ sudo service impala-server stop
   ```
   b. Stop any instances of the state store in your cluster:
   ```
   $ sudo service impala-state-store stop
   ```
   c. Stop any instances of the catalog service in your cluster:
   ```
   $ sudo service impala-catalog stop
   ```
2. Check if there are new recommended or required configuration settings to put into place in the configuration files, typically under `/etc/impala/conf`. See Post-Installation Configuration for Impala on page 221 for settings related to performance and scalability.
3. Use **one** of the following sets of commands to update Impala on each Impala node in your cluster:

   For RHEL, Oracle Linux, or CentOS systems:
   ```
   $ sudo yum update impala-server
   $ sudo zypper update hadoop-lzo-cdh4 # Optional; if this package is already
   installed
   $ sudo yum update impala-catalog # New in Impala 1.2; do yum install when upgrading
    from 1.1.
   ```
   For SUSE systems:
   ```
   $ sudo zypper update impala-server
   $ sudo zypper update hadoop-lzo-cdh4 # Optional; if this package is already
   installed
   $ sudo zypper update impala-catalog # New in Impala 1.2; do zypper install when
   upgrading from 1.1.
   ```
   For Debian or Ubuntu systems:
   ```
   $ sudo apt-get install impala-server
   $ sudo apt-get install hadoop-lzo-cdh4 # Optional; if this package is already
   installed
   $ sudo apt-get install impala-catalog # New in Impala 1.2.
   ```
4. Use **one** of the following sets of commands to update Impala shell on each node on which it is installed:

   For RHEL, Oracle Linux, or CentOS systems:
   ```
   $ sudo yum update impala-shell
   ```
   For SUSE systems:
   ```
   $ sudo zypper update impala-shell
   ```
   For Debian or Ubuntu systems:
   ```
   $ sudo apt-get install impala-shell
   ```

5. Depending on which release of Impala you are upgrading from, you might find that the symbolic links `/etc/impala/conf` and `/usr/lib/impala/sbin` are missing. If so, see Known Issues in the Current Production Release (Impala 1.4.x) for the procedure to work around this problem.
6. Restart Impala services:

   a. Restart the Impala state store service on the desired nodes in your cluster. Expect to see a process named `statestored` if the service started successfully.

   ```
   $ sudo service impala-state-store start
   $ ps ax | grep [s]tatestored
    6819 ?        Sl     0:07 /usr/lib/impala/sbin/statestored
   -log_dir=/var/log/impala -state_store_port=24000
   ```

   Restart the state store service *before* the Impala server service to avoid "Not connected" errors when you run `impala-shell`.

   b. Restart the Impala catalog service on whichever host it runs on in your cluster. Expect to see a process named `catalogd` if the service started successfully.

   ```
   $ sudo service impala-catalog restart
   $ ps ax | grep [c]atalogd
    6068 ?        Sl     4:06 /usr/lib/impala/sbin/catalogd
   ```

   c. Restart the Impala daemon service on each node in your cluster. Expect to see a process named `impalad` if the service started successfully.

   ```
   $ sudo service impala-server start
   $ ps ax | grep [i]mpalad
    7936 ?       Sl     0:12 /usr/lib/impala/sbin/impalad -log_dir=/var/log/impala
    -state_store_port=24000 -use_statestore
   -state_store_host=127.0.0.1 -be_port=22000
   ```

Note:

If the services did not start successfully (even though the `sudo service` command might display `[OK]`), check for errors in the Impala log file, typically in `/var/log/impala`.

Configuring 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 221
- Configuring Impala to Work with ODBC on page 225
- Configuring Impala to Work with JDBC on page 225

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:

- Impala Security Configuration
- Modifying Impala Startup Options on page 227

Post-Installation Configuration for Impala

This section describes the mandatory and recommended configuration settings for Cloudera Impala. If Impala is installed using Cloudera Manager, some of these configurations are completed automatically; you must still configure short-circuit reads manually. If you installed Impala without Cloudera Manager, or if you want to customize your environment, consider making the changes described in this topic.

In some cases, depending on the level of Impala, CDH, and Cloudera Manager, you might need to add particular component configuration details in one of the free-form fields on the Impala configuration pages within Cloudera Manager. In Cloudera Manager 4, these fields are labelled **Safety Valve**; in Cloudera Manager 5, they are called **Advanced Configuration Snippet**.

- You must enable short-circuit reads, whether or not Impala was installed through Cloudera Manager. This setting goes in the Impala configuration settings, not the Hadoop-wide settings.
- If you installed Impala in an environment that is not managed by Cloudera Manager, you must enable block location tracking, and you can optionally enable native checksumming for optimal performance.
- If you deployed Impala using Cloudera Manager see Testing Impala Performance to confirm proper configuration.

Mandatory: Short-Circuit Reads

Enabling short-circuit reads allows Impala to read local data directly from the file system. This removes the need to communicate through the DataNodes, improving performance. This setting also minimizes the number of additional copies of data. Short-circuit reads requires `libhadoop.so` (the Hadoop Native Library) to be accessible to both the server and the client. `libhadoop.so` is not available if you have installed from a tarball. You must install from an `.rpm`, `.deb`, or parcel to use short-circuit local reads.

Note: If you use Cloudera Manager, you can enable short-circuit reads through a checkbox in the user interface and that setting takes effect for Impala as well.

Cloudera strongly recommends using Impala with CDH 4.2 or higher, ideally the latest 4.x release. Impala does support short-circuit reads with CDH 4.1, but for best performance, upgrade to CDH 4.3 or higher. The process of configuring short-circuit reads varies according to which version of CDH you are using. Choose the procedure that is appropriate for your environment.

To configure DataNodes for short-circuit reads with CDH 4.2 or later:

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.

To configure DataNodes for short-circuit reads with CDH 4.1:

> **Note:** Cloudera strongly recommends using Impala with CDH 4.2 or higher, ideally the latest 4.x release. Impala does support short-circuit reads with CDH 4.1, but for best performance, upgrade to CDH 4.3 or higher. The process of configuring short-circuit reads varies according to which version of CDH you are using. Choose the procedure that is appropriate for your environment.

1. Enable short-circuit reads by adding settings to the Impala `core-site.xml` file.
   - If you installed Impala using Cloudera Manager, short-circuit reads should be properly configured, but you can review the configuration by checking the contents of the `core-site.xml` file, which is installed at `/etc/impala/conf` by default.
   - If you installed using packages, instead of using Cloudera Manager, create the `core-site.xml` file. This can be easily done by copying the `core-site.xml` client configuration file from another machine that is running Hadoop services. This file must be copied to the Impala configuration directory. The Impala configuration directory is set by the `IMPALA_CONF_DIR` environment variable and is by default `/etc/impala/conf`. To confirm the Impala configuration directory, check the `IMPALA_CONF_DIR` environment variable value.

     > **Note:** If the Impala configuration directory does not exist, create it and then add the `core-site.xml` file.

   Add the following to the `core-site.xml` file:

   ```
   <property>
      <name>dfs.client.read.shortcircuit</name>
      <value>true</value>
   </property>
   ```

   > **Note:** For an installation managed by Cloudera Manager, specify these settings in the Impala dialogs, in the options field for HDFS. In Cloudera Manager 4, these fields are labelled **Safety Valve**; in Cloudera Manager 5, they are called **Advanced Configuration Snippet**.

2. For each DataNode, enable access by adding the following to the `hdfs-site.xml` file:

   ```
   <property>
      <name>dfs.client.use.legacy.blockreader.local</name>
      <value>true</value>
   </property>

   <property>
      <name>dfs.datanode.data.dir.perm</name>
      <value>750</value>
   </property>

   <property>
      <name>dfs.block.local-path-access.user</name>
      <value>impala</value>
   </property>

   <property>
       <name>dfs.client.file-block-storage-locations.timeout.millis</name>
       <value>10000</value>
   </property>
   ```

   > **Note:** In the preceding example, the `dfs.block.local-path-access.user` is the user running the `impalad` process. By default, that account is `impala`.

3. Use `usermod` to add users requiring local block access to the appropriate HDFS group. For example, if you assigned `impala` to the `dfs.block.local-path-access.user` property, you would add `impala` to the hadoop HDFS group:

   ```
   $ usermod -a -G hadoop impala
   ```

   > **Note:** The default HDFS group is hadoop, but it is possible to have an environment configured to use an alternate group. To find the configured HDFS group name using the Cloudera Manager admin console, click **Services** and click **HDFS**. Click the **Configuration** tab. Under **Service-Wide**, click **Advanced** in the left column. The **Shared Hadoop Group Name** property contains the group name.

   > **Note:** If you are going to enable block location tracking, you can skip copying configuration files and restarting DataNodes and go straight to Mandatory: Block Location Tracking on page 224. 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. 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`.
5. 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.

> **Note:** You may need to sign in and accept license agreements before accessing the pages required for downloading ODBC connectors.

Version 1.x of the Cloudera ODBC Connectors uses the original HiveServer1 protocol, corresponding to Impala port 21000.

The newer versions 2.5 and 2.0, 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.

See the downloads page for the versions of these drivers for different products, and the documentation page for installation instructions.

> **Important:** If you are using the Cloudera Connector for Tableau, to connect Impala to your Kerberos-secured CDH clusters, contact your Tableau account representative for an updated Tableau Data-connection Customization (TDC) file. The updated TDC file will override the Tableau connection settings to set specific parameters on the connection string that are required for a secure connection.

Configuring Impala to Work with JDBC

Impala supports JDBC integration. 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:

- Specifying an available communication port. See Configuring the JDBC Port on page 225.
- Installing the JDBC driver on every system that runs the JDBC-enabled application. See Enabling Impala JDBC Support on Client Systems on page 225.
- Specifying a connection string for the JDBC application to access one of the servers running the `impalad` daemon, with the appropriate security settings. See Establishing JDBC Connections on page 226.

Configuring the JDBC Port

The default port used by JDBC 2.0 (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 on page 227 for details.

Enabling Impala JDBC Support on Client Systems

The Impala JDBC integration is made possible by a client-side JDBC driver, made up of some Java JAR files. The same driver is used by Impala and Hive. To get the JAR files, install the Hive JDBC driver on one machine by following the instructions for CDH 4 or CDH 5. 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
```

```
httpcore-X.X.X.jar
httpclient-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 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
```

Starting Impala

To begin using Cloudera Impala:

1. Set any necessary configuration options for the Impala services. See Modifying Impala Startup Options on page 227 for details.
2. Start one instance of the Impala statestore. The statestore helps Impala to distribute work efficiently, and to continue running in the event of availability problems for other Impala nodes. If the statestore becomes unavailable, Impala continues to function.
3. Start one instance of the Impala catalog service.
4. Start the main Impala service on one or more DataNodes, ideally on all DataNodes to maximize local processing and avoid network traffic due to remote reads.

Once Impala is running, you can conduct interactive experiments using the instructions in Impala Tutorial and try Using the Impala Shell (impala-shell Command).

Starting Impala through Cloudera Manager

If you installed Impala with Cloudera Manager, use Cloudera Manager to start and stop services. The Cloudera Manager GUI is a convenient way to check that all services are running, to set configuration options using form fields in a browser, and to spot potential issues such as low disk space before they become serious. Cloudera Manager automatically starts all the Impala-related services as a group, in the correct order. See the Cloudera Manager Documentation for details.

Starting Impala from the Command Line

To start the Impala state store and Impala from the command line or a script, you can either use the `service` command or you can start the daemons directly through the `impalad`, `statestored`, and `catalogd` executables.

Start the Impala statestore and then start `impalad` instances. You can modify the values the service initialization scripts use when starting the statestore and Impala by editing `/etc/default/impala`.

Start the statestore service using a command similar to the following:

```
$ sudo service impala-state-store start
```

Start the catalog service using a command similar to the following:

```
$ sudo service impala-catalog start
```

Start the Impala service on each data node using a command similar to the following:

```
$ sudo service impala-server start
```

If any of the services fail to start, review:

- Reviewing Impala Logs
- Appendix B - Troubleshooting Impala

Modifying Impala Startup Options

The configuration options for the Impala-related daemons let you choose which hosts and ports to use for the services that run on a single host, specify directories for logging, control resource usage and security, and specify other aspects of the Impala software.

Configuring Impala Startup Options through Cloudera Manager

If you manage your cluster through Cloudera Manager, configure the settings for all the Impala-related daemons by navigating to this page: **Services** > **Impala** > **Configuration** > **View and Edit**. See the Cloudera Manager documentation for instructions about how to configure Impala through Cloudera Manager: Impala information for Cloudera Manager 4, Impala information for Cloudera Manager 5.

If the Cloudera Manager interface does not yet have a form field for a newly added option, or if you need to use special options for debugging and troubleshooting, the **Advanced** option page for each daemon includes one or more fields where you can enter option names directly. In Cloudera Manager 4, these fields are labelled **Safety Valve**; in Cloudera Manager 5, they are called **Advanced Configuration Snippet**. There is also a free-form field for query options, on the top-level **Impala Daemon** options page.

Configuring Impala Startup Options through the Command Line

When you run Impala in a non-Cloudera Manager environment, the Impala server, statestore, and catalog services start up using values provided in a defaults file, `/etc/default/impala`.

This file includes information about many resources used by Impala. Most of the defaults included in this file should be effective in most cases. For example, typically you would not change the definition of the CLASSPATH variable, but you would always set the address used by the statestore server. Some of the content you might modify includes:

```
IMPALA_STATE_STORE_HOST=127.0.0.1
IMPALA_STATE_STORE_PORT=24000
IMPALA_BACKEND_PORT=22000
IMPALA_LOG_DIR=/var/log/impala
IMPALA_CATALOG_SERVICE_HOST=...
IMPALA_STATE_STORE_HOST=...

export IMPALA_STATE_STORE_ARGS=${IMPALA_STATE_STORE_ARGS:- \
    -log_dir=${IMPALA_LOG_DIR} -state_store_port=${IMPALA_STATE_STORE_PORT}}
IMPALA_SERVER_ARGS=" \
-log_dir=${IMPALA_LOG_DIR} \
-catalog_service_host=${IMPALA_CATALOG_SERVICE_HOST} \
-state_store_port=${IMPALA_STATE_STORE_PORT} \
-use_statestore \
-state_store_host=${IMPALA_STATE_STORE_HOST} \
-be_port=${IMPALA_BACKEND_PORT}"
export ENABLE_CORE_DUMPS=${ENABLE_COREDUMPS:-false}
```

To use alternate values, edit the defaults file, then restart all the Impala-related services so that the changes take effect. Restart the Impala server using the following commands:

```
$ sudo service impala-server restart
Stopping Impala Server:                                    [  OK  ]
Starting Impala Server:                                    [  OK  ]
```

Restart the Impala statestore using the following commands:

```
$ sudo service impala-state-store restart
Stopping Impala State Store Server:                        [  OK  ]
Starting Impala State Store Server:                        [  OK  ]
```

Restart the Impala catalog service using the following commands:

```
$ sudo service impala-catalog restart
Stopping Impala Catalog Server:                            [  OK  ]
Starting Impala Catalog Server:                            [  OK  ]
```

Some common settings to change include:

- Statestore address. Cloudera recommends the statestore be on a separate host not running the `impalad` daemon. In that recommended configuration, the `impalad` daemon cannot refer to the statestore server using the loopback address. If the statestore is hosted on a machine with an IP address of 192.168.0.27, change:

```
IMPALA_STATE_STORE_HOST=127.0.0.1
```

to:

```
IMPALA_STATE_STORE_HOST=192.168.0.27
```

- Catalog server address (including both the hostname and the port number). Update the value of the `IMPALA_CATALOG_SERVICE_HOST` variable. Cloudera recommends the catalog server be on the same host as the statestore. In that recommended configuration, the `impalad` daemon cannot refer to the catalog server using the loopback address. If the catalog service is hosted on a machine with an IP address of 192.168.0.27, add the following line:

```
IMPALA_CATALOG_SERVICE_HOST=192.168.0.27:26000
```

The `/etc/default/impala` defaults file currently does not define an `IMPALA_CATALOG_ARGS` environment variable, but if you add one it will be recognized by the service startup/shutdown script. Add a definition for this variable to `/etc/default/impala` and add the option `-catalog_service_host=`*`hostname`*. If the port is different than the default 26000, also add the option `-catalog_service_port=`*`port`*.

- Memory limits. You can limit the amount of memory available to Impala. For example, to allow Impala to use no more than 70% of system memory, change:

```
export IMPALA_SERVER_ARGS=${IMPALA_SERVER_ARGS:- \
    -log_dir=${IMPALA_LOG_DIR} \
    -state_store_port=${IMPALA_STATE_STORE_PORT} \
    -use_statestore -state_store_host=${IMPALA_STATE_STORE_HOST} \
    -be_port=${IMPALA_BACKEND_PORT}}
```

to:

```
export IMPALA_SERVER_ARGS=${IMPALA_SERVER_ARGS:- \
    -log_dir=${IMPALA_LOG_DIR} -state_store_port=${IMPALA_STATE_STORE_PORT} \
    -use_statestore -state_store_host=${IMPALA_STATE_STORE_HOST} \
    -be_port=${IMPALA_BACKEND_PORT} -mem_limit=70%}
```

You can specify the memory limit using absolute notation such as `500m` or `2G`, or as a percentage of physical memory such as `60%`.

Note: Queries that exceed the specified memory limit are aborted. Percentage limits are based on the physical memory of the machine and do not consider cgroups.

- Core dump enablement. To enable core dumps, change:

```
export ENABLE_CORE_DUMPS=${ENABLE_COREDUMPS:-false}
```

to:

```
export ENABLE_CORE_DUMPS=${ENABLE_COREDUMPS:-true}
```

Note: The location of core dump files may vary according to your operating system configuration. Other security settings may prevent Impala from writing core dumps even when this option is enabled.

- Authorization using the open source Sentry plugin. Specify the `-server_name` and `-authorization_policy_file` options as part of the `IMPALA_SERVER_ARGS` and `IMPALA_STATE_STORE_ARGS`

settings to enable the core Impala support for authentication. See Starting the impalad Daemon with Sentry Authorization Enabled for details.
- Auditing for successful or blocked Impala queries, another aspect of security. Specify the `-audit_event_log_dir=directory_path` option and optionally the `-max_audit_event_log_file_size=number_of_queries` and `-abort_on_failed_audit_event` options as part of the `IMPALA_SERVER_ARGS` settings, for each Impala node, to enable and customize auditing. See Auditing Impala Operations for details.
- Password protection for the Impala web UI, which listens on port 25000 by default. This feature involves adding some or all of the `--webserver_password_file`, `--webserver_authentication_domain`, and `--webserver_certificate_file` options to the `IMPALA_SERVER_ARGS` and `IMPALA_STATE_STORE_ARGS` settings. See Security Guidelines for Impala for details.
- Another setting you might add to `IMPALA_SERVER_ARGS` is:

  ```
  -default_query_options='option=value;option=value;...'
  ```

  These options control the behavior of queries performed by this `impalad` instance. The option values you specify here override the default values for Impala query options, as shown by the `SET` command in `impala-shell`.
- Options for resource management, in conjunction with the YARN and Llama components. These options include `-enable_rm`, `-llama_host`, `-llama_port`, `-llama_callback_port`, and `-cgroup_hierarchy_path`. Additional options to help fine-tune the resource estimates are `-—rm_always_use_defaults`, `-—rm_default_memory=size`, and `-—rm_default_cpu_cores`. For details about these options, see impalad Startup Options for Resource Management. See Using YARN Resource Management with Impala (CDH 5 Only) for information about resource management in general, and The Llama Daemon for information about the Llama daemon.
- During troubleshooting, Cloudera Support might direct you to change other values, particularly for `IMPALA_SERVER_ARGS`, to work around issues or gather debugging information.

The following startup options for `impalad` enable resource management and customize its parameters for your cluster configuration:

- `-enable_rm`: Whether to enable resource management or not, either `true` or `false`. The default is `false`. None of the other resource management options have any effect unless `-enable_rm` is turned on.
- `-llama_host`: Hostname or IP address of the Llama service that Impala should connect to. The default is `127.0.0.1`.
- `-llama_port`: Port of the Llama service that Impala should connect to. The default is 15000.
- `-llama_callback_port`: Port that Impala should start its Llama callback service on. Llama reports when resources are granted or preempted through that service.
- `-cgroup_hierarchy_path`: Path where YARN and Llama will create CGroups for granted resources. Impala assumes that the CGroup for an allocated container is created in the path '*cgroup_hierarchy_path* + *container_id*'.
- `-rm_always_use_defaults`: If this Boolean option is enabled, Impala ignores computed estimates and always obtains the default memory and CPU allocation from Llama at the start of the query. These default estimates are approximately 2 CPUs and 4 GB of memory, possibly varying slightly depending on cluster size, workload, and so on. Cloudera recommends enabling `-rm_always_use_defaults` whenever resource management is used, and relying on these default values (that is, leaving out the two following options).
- `-rm_default_memory=size`: Optionally sets the default estimate for memory usage for each query. You can use suffixes such as MB and GB, MEM_LIMIT query option. Only has an effect when `-rm_always_use_defaults` is also enabled.
- `-rm_default_cpu_cores`: Optionally sets the default estimate for number of virtual CPU cores for each query. Only has an effect when `-rm_always_use_defaults` is also enabled.

Note:

These startup options for the `impalad` daemon are different from the command-line options for the `impala-shell` command. For the `impala-shell` options, see impala-shell Command-Line Options.

Checking the Values of Impala Configuration Options

You can check the current runtime value of all these settings through the Impala web interface, available by default at `http://`*`impala_hostname`*`:25000/varz` (impalad) `http://`*`impala_hostname`*`:25010/varz` (`statestored`), or `http://`*`impala_hostname`*`:25020/varz` (`catalogd`). In the Cloudera Manager interface, you can see the link to the appropriate ***service_name*** **Web UI** page when you look at the status page for a specific daemon on a specific host.

Startup Options for impalad Daemon

The `impalad` daemon implements the main Impala service, which performs query processing and reads and writes the data files.

Startup Options for statestored Daemon

The `statestored` daemon implements the Impala statestore service, which monitors the availability of Impala services across the cluster, and handles situations such as nodes becoming unavailable or becoming available again.

Startup Options for catalogd Daemon

The `catalogd` daemon implements the Impala catalog service, which broadcasts metadata changes to all the Impala nodes when Impala creates a table, inserts data, or performs other kinds of DDL and DML operations.

By default, the metadata loading and caching on startup happens asynchronously, so Impala can begin accepting requests promptly. To enable the original behavior, where Impala waited until all metadata was loaded before accepting any requests, set the `catalogd` configuration option `--load_catalog_in_background=false`.

Hive Installation

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper/YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation and the instructions for upgrading to CDH 5 .

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

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

- About Hive
- Upgrading Hive
- Installing Hive
- Configuring the Metastore
- Configuring HiveServer2
- Starting the Metastore
- File System Permissions
- Starting, Stopping, and Using HiveServer2
- Using Hive with HBase
- Using the Hive Schema Tool on page 254
- Using the Hive Schema Tool on page 254
- Installing the JDBC on Clients
- Setting HADOOP_MAPRED_HOME on page 257
- Configuring the Metastore for HDFS HA
- Troubleshooting
- Apache Hive Documentation

About Hive

Apache Hive is a powerful data warehousing application built on top of Hadoop; it enables you to access your data using Hive QL, a language that is similar to SQL.

Note:

As of CDH 5, Hive includes HCatalog, but you still need to install HCatalog separately if you want to use it; see Installing and Using HCatalog on page 203.

Install Hive on your client machine(s) from which you submit jobs; you do not need to install it on the nodes in your Hadoop cluster.

HiveServer2

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

> **Important:**
>
> The original HiveServer and command-line interface (CLI) are no longer supported; use HiveServer2 and Beeline.

Upgrading Hive

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

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

Checklist to Help Ensure Smooth Upgrades

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

- Configure periodic backups of the metastore database. Use `mysqldump`, or the equivalent for your vendor if you are not using MySQL.
- Make sure `datanucleus.autoCreateSchema` is set to false (in all types of database) and `datanucleus.fixedDatastore` is set to true (for MySQL and Oracle) in *all* `hive-site.xml` files. See the configuration instructions for more information about setting the properties in `hive-site.xml`.
- Insulate the metastore database from users by running the metastore service in Remote mode. If you do not follow this recommendation, make sure you remove `DROP`, `ALTER`, and `CREATE` privileges from the Hive user configured in `hive-site.xml`. See Configuring the Hive Metastore on page 238 for complete instructions for each type of supported database.

Upgrading Hive from CDH 4 to CDH 5

> **Note:**
>
> If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5, you can skip Step 1 below and proceed with installing the new CDH 5 version of Hive.

Step 1: Remove Hive

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

1. Exit the Hive console and make sure no Hive scripts are running.
2. Stop any HiveServer processes that are running. If HiveServer is running as a daemon, use the following command to stop it:

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

   If HiveServer is running from the command line, stop it with <CTRL>-c.

3. Stop the metastore. If the metastore is running as a daemon, use the following command to stop it:

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

   If the metastore is running from the command line, stop it with <CTRL>-c.

4. Remove Hive:

   ```
   $ sudo yum remove hive
   ```

   To remove Hive on SLES systems:

   ```
   $ sudo zypper remove hive
   ```

   To remove Hive on Ubuntu and Debian systems:

   ```
   $ sudo apt-get remove hive
   ```

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

See Installing Hive.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Step 3: Configure the Hive Metastore

You must configure the Hive metastore and initialize the service before you start the Hive Console. See Configuring the Hive Metastore for detailed instructions.

Step 4: Upgrade the Metastore Schema

Important:

- Cloudera strongly encourages you to make a backup copy of your metastore database before running the upgrade scripts. You will need this backup copy if you run into problems during the upgrade or need to downgrade to a previous version.
- You **must** upgrade the metastore schema before starting Hive after the upgrade. Failure to do so may result in metastore corruption.
- To run a script, you must first cd to the directory that script is in: that is `/usr/lib/hive/scripts/metastore/upgrade/<database>`.

The current version of CDH 5 includes changes in the Hive metastore schema. If you have been using Hive 0.10 or earlier, you must upgrade the Hive metastore schema after you install the new version of Hive but before you start Hive.

With CDH 5, there are now two ways to do this. You could either use Hive's `schematool` or use the schema upgrade scripts available with the Hive package.

Using `schematool` (Recommended):

The Hive distribution now includes an offline tool for Hive metastore schema manipulation called `schematool`. This tool can be used to initialize the metastore schema for the current Hive version. It can also handle upgrading schema from an older version to the current one. To do this, use the `upgradeSchemaFrom` option to specify the version of the schema you are currently using (see table below) and the compulsory `dbType` option to specify the database you are using. For example,

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

Possible values for the `dbType` option are `mysql`, `postgres`, `derby` or `oracle`. The following table lists the Hive versions corresponding to the older CDH releases.

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

See Using the Hive Schema Tool for more details on how to use `schematool`.

Using Schema Upgrade Scripts:

Run the appropriate schema upgrade scripts available in `/usr/lib/hive/scripts/metastore/upgrade/`:

- Schema upgrade scripts from 0.7 to 0.8 and from 0.8 to 0.9 for Derby, MySQL, and PostgreSQL
- 0.8 and 0.9 schema scripts for Oracle, but no upgrade scripts (you will need to create your own)
- Schema upgrade scripts from 0.9 to 0.10 for Derby, MySQL, PostgreSQL and Oracle
- Schema upgrade scripts from 0.10 to 0.11 for Derby, MySQL, PostgreSQL and Oracle

For more information about upgrading the schema, see the README in `/usr/lib/hive/scripts/metastore/upgrade/`.

Step 5: Configure HiveServer2

HiveServer2 is an improved version of the original HiveServer (HiveServer1, no longer supported). Some configuration is required before you initialize HiveServer2; see Configuring HiveServer2 for details.

Step 6: Upgrade Scripts, etc., for HiveServer2 (if necessary)

If you have been running HiveServer1, you may need to make some minor modifications to your client-side scripts and applications when you upgrade:

- HiveServer1 does not support concurrent connections, so many customers run a dedicated instance of HiveServer1 for each client. These can now be replaced by a single instance of HiveServer2.
- HiveServer2 uses a different connection URL and driver class for the JDBC driver. If you have existing scripts that use JDBC to communicate with HiveServer1, you can modify these scripts to work with HiveServer2 by

changing the JDBC driver URL from `jdbc:hive://hostname:port` to `jdbc:hive2://hostname:port`, and by changing the JDBC driver class name from `org.apache.hive.jdbc.HiveDriver` to `org.apache.hive.jdbc.HiveDriver`.

Step 7: Start the Metastore, HiveServer2, and Beeline

See:

- Starting the Metastore
- Starting HiveServer2
- Using Beeline

Step 8: Upgrade the JDBC driver on the clients

The driver used for CDH 4.*x* does not work with CDH 5.*x*. Install the new version, following these instructions.

Upgrading Hive from an Earlier Version of CDH 5

The instructions that follow assume that you are upgrading Hive as part of a CDH 5 upgrade, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

Important:

- If you are currently running Hive under MRv1, check for the following property and value in `/etc/mapred/conf/mapred-site.xml`:

  ```
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
  </property>
  ```

  Remove this property before you proceed; otherwise Hive queries spawned from MapReduce jobs will fail with a null pointer exception (NPE).
- If you have installed the `hive-hcatalog-server` package in the past, you must remove it before you proceed; otherwise the upgrade will fail.

To upgrade Hive from an earlier version of CDH 5, proceed as follows.

Step 1: Stop all Hive Processes and Daemons

Warning:

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

1. Stop any HiveServer processes that are running:

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

2. Stop any HiveServer2 processes that are running:

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

3. Stop the metastore:

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

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

SeeInstalling Hive on page 238

Step 3: Verify that the Hive Metastore is Properly Configured

See Configuring the Hive Metastore on page 238 for detailed instructions.

Step 4: Start the Metastore, HiveServer2, and Beeline

See:

- Starting the Metastore on page 252
- Starting, Stopping, and Using HiveServer2 on page 252

The upgrade is now complete.

Installing Hive

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

RedHat and CentOS systems	`$ sudo yum install <pkg1> <pkg2> ...`
SLES systems	`$ sudo zypper install <pkg1> <pkg2> ...`
Ubuntu and Debian systems	`$ sudo apt-get install <pkg1> <pkg2> ...`

The packages are:

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

Configuring the Hive Metastore

The Hive metastore service stores the metadata for Hive tables and partitions in a relational database, and provides clients (including Hive) access to this information via the metastore service API. The subsections that follow discuss the deployment options and provide instructions for setting up a database in a recommended configuration.

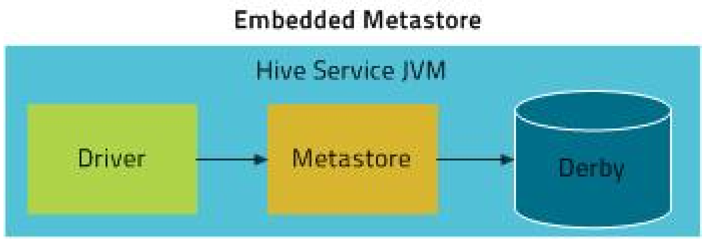
Metastore Deployment Modes

Note:

HiveServer in the discussion that follows refers to HiveServer1 or HiveServer2, whichever you are using.

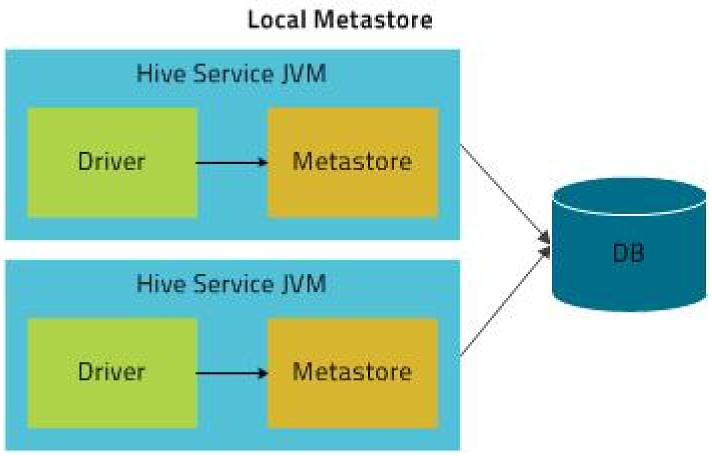
Embedded Mode

Cloudera recommends using this mode for experimental purposes only.



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

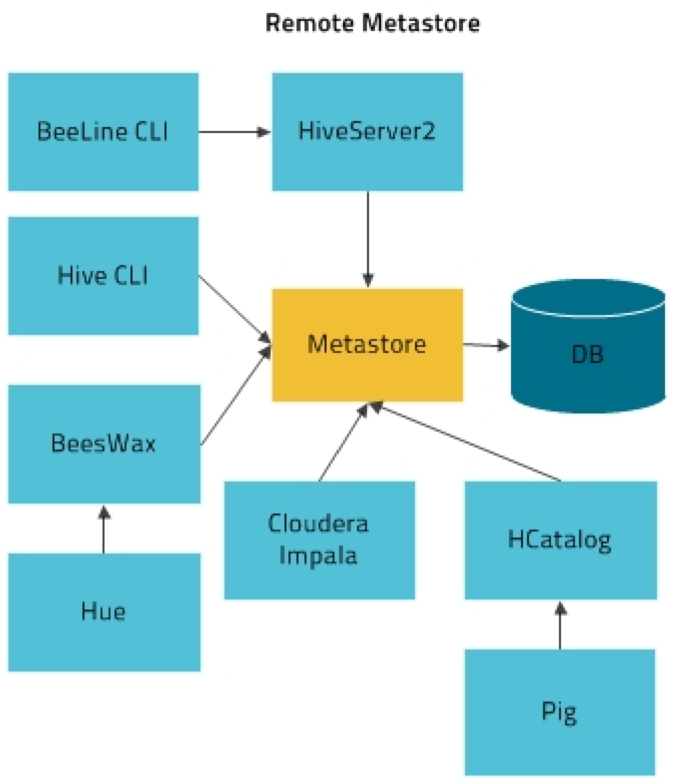
Local Mode



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

Remote Mode

Cloudera recommends that you use this mode.



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

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

Supported Metastore Databases

See the CDH 5 Requirements and Supported Versions page for up-to-date information on supported databases. Cloudera strongly encourages you to use MySQL because it is the most popular with the rest of the Hive user community, and so receives more testing than the other options.

Configuring the Metastore Database

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

The configuration properties for the Hive metastore are documented on the Hive Metastore documentation page, which also includes a pointer to the E/R diagram for the Hive metastore.

Note:

For information about additional configuration that may be needed in a secure cluster, see Hive Security Configuration.

Configuring a remote MySQL database for the Hive Metastore

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

Step 1: Install and start MySQL if you have not already done so

To install MySQL on a Red Hat system:

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

To install MySQL on a SLES system:

```
$ sudo zypper install mysql
$ sudo zypper install libmysqlclient_r15
```

To install MySQL on an Debian/Ubuntu system:

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

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

On Red Hat systems

```
$ sudo service mysqld start
```

On SLES and Debian/Ubuntu systems

```
$ sudo service mysql start
```

Step 2: Configure the MySQL Service and Connector

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

To install the MySQL connector on a Red Hat 6 system:

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

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

To install the MySQL connector on a Red Hat 5 system:

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

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

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

To install the MySQL connector on a SLES system:

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

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

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

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

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

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

To set the MySQL root password:

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

To make sure the MySQL server starts at boot:

- On Red Hat systems:

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

- On SLES systems:

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

- On Debian/Ubuntu systems:

```
$ sudo chkconfig mysql on
```

Step 3. Create the Database and User

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

Note:

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

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

If for some reason you decide not to use the schema tool, you can use the `hive-schema-0.12.0.mysql.sql` file instead; that file is located in the `/usr/lib/hive/scripts/metastore/upgrade/mysql` directory. Proceed as follows if you decide to use `hive-schema-0.12.0.mysql.sql`.

Example using hive-schema-0.12.0.mysql.sql

Note:

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

```
$ mysql -u root -p
Enter password:
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE /usr/lib/hive/scripts/metastore/upgrade/mysql/hive-schema-0.12.0.mysql.sql;
```

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

Important:

If you fail to restrict the ability of the metastore MySQL user account to create and alter tables, it is possible that users will inadvertently corrupt the metastore schema when they use older or newer versions of Hive.

Example

```
mysql> CREATE USER 'hive'@'metastorehost' IDENTIFIED BY 'mypassword';
...
mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'hive'@'metastorehost';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,LOCK TABLES,EXECUTE ON metastore.* TO
'hive'@'metastorehost';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

Step 4: Configure the Metastore Service to Communicate with the MySQL Database

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

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

Note:

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

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

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

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

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

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

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

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

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

Configuring a remote PostgreSQL database for the Hive Metastore

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

Step 1: Install and start PostgreSQL if you have not already done so

To install PostgreSQL on a Red Hat system:

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

To install PostgreSQL on a SLES system:

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

To install PostgreSQL on an Debian/Ubuntu system:

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

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

To initialize database files on Red Hat compatible systems

```
$ sudo service postgresql initdb
```

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

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

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

On Red-Hat-compatible systems:

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

On SLES systems:

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

On Ubuntu and Debian systems:

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

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

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

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

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

Note:

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

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

Start PostgreSQL Server

```
$ sudo service postgresql start
```

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

```
chkconfig postgresql on
```

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

```
chkconfig --list postgresql
```

Step 2: Install the Postgres JDBC Driver

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

To install the PostgreSQL JDBC Driver on a Red Hat 6 system:

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

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

To install the PostgreSQL connector on a Red Hat 5 system:

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

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

Note:

You may need to use a different version if you have a different version of Postgres. You can check the version as follows:

```
$ sudo rpm -qa | grep postgres
```

To install the PostgreSQL JDBC Driver on a SLES system:

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

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

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

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

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

Step 3: Create the metastore database and user account

Proceed as in the following example, using the appropriate script in `/usr/lib/hive/scripts/metastore/upgrade/postgres/`:

```
$ sudo -u postgres psql
postgres=# CREATE USER hiveuser WITH PASSWORD 'mypassword';
postgres=# CREATE DATABASE metastore;
postgres=# \c metastore;
You are now connected to database 'metastore'.
postgres=# \i
/usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schema-0.12.0.postgres.sql
SET
SET
...
```

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

Note:

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

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

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

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

Step 4: Configure the Metastore Service to Communicate with the PostgreSQL Database

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

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

Note:

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

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

```
</property>

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

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

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

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

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

Step 6: Test connectivity to the metastore:

```
$ hive –e “show tables;”
```

Note:

This will take a while the first time.

Configuring a remote Oracle database for the Hive Metastore

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

Step 1: Install and start Oracle

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

Step 2: Install the Oracle JDBC Driver

You must download the Oracle JDBC Driver from the Oracle website and put the file `ojdbc6.jar` into `/usr/lib/hive/lib/` directory. The driver is available for download here.

Note:

These URLs were correct at the time of publication, but the Oracle site is restructured frequently.

```
$ sudo mv ojdbc6.jar /usr/lib/hive/lib/
```

Step 3: Create the Metastore database and user account

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

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

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

```
$ sqlplus hiveuser
SQL> @/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-0.12.0.oracle.sql
```

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

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

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

Step 4: Configure the Metastore Service to Communicate with the Oracle Database

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

Example

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

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

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

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

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

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

<property>
```

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

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

Configuring HiveServer2

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

Table Lock Manager (Required)

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

Important:

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

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

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

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

Important:

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

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

hive.zookeeper.client.port

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

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

JDBC driver

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

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

Authentication

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

Running HiveServer2 and HiveServer Concurrently

Important:

Cloudera strongly recommends running HiveServer2 instead of the original HiveServer (HiveServer1) package; HiveServer1 is deprecated.

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

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

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

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

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

Starting the Metastore

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

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

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

File System Permissions

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

> **Important:**
>
> Cloudera recommends setting permissions on the Hive warehouse directory to `1777`, making it accessible to all users, with the sticky bit set. This allows users to create and access their tables, but prevents them from deleting tables they don't own.

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

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

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

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

Starting, Stopping, and Using HiveServer2

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

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

To start HiveServer2:

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

To stop HiveServer2:

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

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

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

Using the Beeline CLI

Beeline is the CLI (command-line interface) for HiveServer2. It is based on the SQLLine CLI written by Marc Prud'hommeaux.

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

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

Note:

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

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

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

Starting HiveServer1 and the Hive Console

Important:

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

To start HiveServer1:

```
$ sudo service hiveserver start
```

See also Running HiveServer2 and HiveServer Concurrently on page 251.

To start the Hive console:

```
$ hive
hive>
```

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

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

Using Hive with HBase

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

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

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

Using the Hive Schema Tool

Schema Version Verification

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

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

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

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

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

Using schematool

The Hive distribution now includes an offline tool for Hive metastore schema manipulation called `schematool`. This tool can be used to initialize the metastore schema for the current Hive version. It can also handle upgrading schema from an older version to the current one. The tool will try to find the current schema from the metastore if available. However, this will be applicable only to any future upgrades. In case you are upgrading from existing CDH releases like CDH 4 or CDH 3, you should specify the schema version of the existing metastore as a command line option to the tool.

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

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

```
$ schematool -help
usage: schemaTool
 -dbType <databaseType>             Metastore database type
 -dryRun                            List SQL scripts (no execute)
 -help                              Print this message
 -info                              Show config and schema details
 -initSchema                        Schema initialization
 -initSchemaTo <initTo>             Schema initialization to a version
 -passWord <password>               Override config file password
 -upgradeSchema                     Schema upgrade
 -upgradeSchemaFrom <upgradeFrom>   Schema upgrade from a version
 -userName <user>                   Override config file user name
 -verbose                           Only print SQL statements
```

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

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

Usage Examples

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

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

- Get schema information using the `info` option.

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

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

```
$ schematool -dbType derby -info
Metastore connection URL:       jdbc:derby:;databaseName=metastore_db;create=true
Metastore Connection Driver :    org.apache.derby.jdbc.EmbeddedDriver
Metastore connection User:       APP
```

```
Hive distribution version:       <new_version>
Required schema version:         <new_version>
org.apache.hadoop.hive.metastore.HiveMetaException: Failed to get schema version.
*** schemaTool failed ***
```

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

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

The Hive versions of the older CDH releases are:

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

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

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

Installing the Hive JDBC on Clients

If you want to install only the JDBC on your Hive clients, proceed as follows.

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

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

   - On SLES systems:

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

- On Ubuntu or Debian systems:

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

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

You are now ready to run your JDBC client. For more information see the Hive Client document.

Setting HADOOP_MAPRED_HOME

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

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

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

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

Configuring the Metastore to use HDFS High Availability

See Upgrading the Hive Metastore to use HDFS HA.

Troubleshooting

This section provides guidance on problems you may encounter while installing, upgrading, or running Hive.

Hive Queries Fail with "Too many counters" Error

Explanation

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

Note:

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

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

What To Do

If you run into this error, set `mapreduce.job.counters.max` in `mapred-site.xml` to a higher value.

Viewing the Hive Documentation

For additional Hive documentation, see the Apache Hive wiki.

Hive Installation

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

HttpFS Installation

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Use the following sections to install and configure HttpFS:

- About HttpFS
- Packaging
- Prerequisites
- Installing HttpFS
- Configuring HttpFS
- Starting the Server
- Stopping the Server
- Using the Server with curl

About HttpFS

Apache Hadoop HttpFS is a service that provides HTTP access to HDFS.

HttpFS has a REST HTTP API supporting all HDFS File System 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 via a plugin API. HttpFS also supports Hadoop proxy user functionality.

The `webhdfs` client file system implementation can access HttpFS via 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 http://archive.cloudera.com/cdh5/cdh/5/hadoop/hadoop-hdfs-httpfs/index.html.

HttpFS Packaging

There are two packaging options for installing HttpFS:

- The `hadoop-httpfs` RPM package

- The hadoop-`httpfs` Debian package

You can also download a Hadoop tarball, which includes HttpFS, from here.

HttpFS Prerequisites

Prerequisites for installing HttpFS are:

- A Unix-like system: see CDH 5 Requirements and Supported Versions for details
- Java: see Java Development Kit Installation for details

Note:

To see which version of HttpFS is shipping in CDH 5, check the Version and Packaging Information. For important information on new and changed components, see the CDH 5 Release Notes. CDH 5 Hadoop works with the CDH 5 version of HttpFS.

Installing HttpFS

HttpFS is distributed in the hadoop-`httpfs` package. To install it, use your preferred package manager application. Install the package on the system that will run the HttpFS server.

Important:

If you have not already done so, install Cloudera's Yum, zypper/YaST or Apt repository before using the following commands to install HttpFS. For instructions, see CDH 5 Installation on page 27.

To install the HttpFS package on a Red Hat-compatible system:

```
$ sudo yum install hadoop-httpfs
```

To install the HttpFS server package on a SLES system:

```
$ sudo zypper install hadoop-httpfs
```

To install the HttpFS package on an Ubuntu or Debian system:

```
$ sudo apt-get install hadoop-httpfs
```

Note:

Installing the `httpfs` package creates an `httpfs` service configured to start HttpFS at system startup time.

You are now ready to configure HttpFS. See the next section.

Configuring HttpFS

When you install HttpFS from an RPM or Debian package, HttpFS creates all configuration, documentation, and runtime files in the standard Unix directories, as follows.

Type of File	Where Installed
Binaries	`/usr/lib/hadoop-httpfs/`
Configuration	`/etc/hadoop-httpfs/conf/`
Documentation	*for SLES:* `/usr/share/doc/packages/hadoop-httpfs/`
	for other platforms: `/usr/share/doc/hadoop-httpfs/`
Data	`/var/lib/hadoop-httpfs/`
Logs	`/var/log/hadoop-httpfs/`
temp	`/var/tmp/hadoop-httpfs/`
PID file	`/var/run/hadoop-httpfs/`

Configuring the HDFS HttpFS Will Use

HttpFS reads the HDFS configuration from the `core-site.xml` and `hdfs-site.xml` files in `/etc/hadoop/conf/`. If necessary edit those files to configure the HDFS HttpFS will use.

Configuring the HttpFS Proxy User

Edit `core-site.xml` and define the Linux user that will run the HttpFS server as a Hadoop proxy user. For example:

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

Then restart Hadoop to make the proxy user configuration active.

Configuring HttpFS with Kerberos Security

To configure HttpFS with Kerberos Security, see HttpFS Security Configuration.

Starting the HttpFS Server

After you have completed all of the required configuration steps, you can start HttpFS:

```
$ sudo service hadoop-httpfs start
```

If you see the message `Server httpfs started!, status NORMAL` in the `httpfs.log` log file, the system has started successfully.

Note:

By default, HttpFS server runs on port 14000 and its URL is `http://<HTTPFS_HOSTNAME>:14000/webhdfs/v1`.

Stopping the HttpFS Server

To stop the HttpFS server:

```
$ sudo service hadoop-httpfs stop
```

Using the HttpFS Server with curl

You can use a tool such as `curl` to access HDFS via HttpFS. For example, to obtain the home directory of the user babu, use a command such as this:

```
$ curl "http://localhost:14000/webhdfs/v1?op=gethomedirectory&user.name=babu"
```

You should see output such as this:

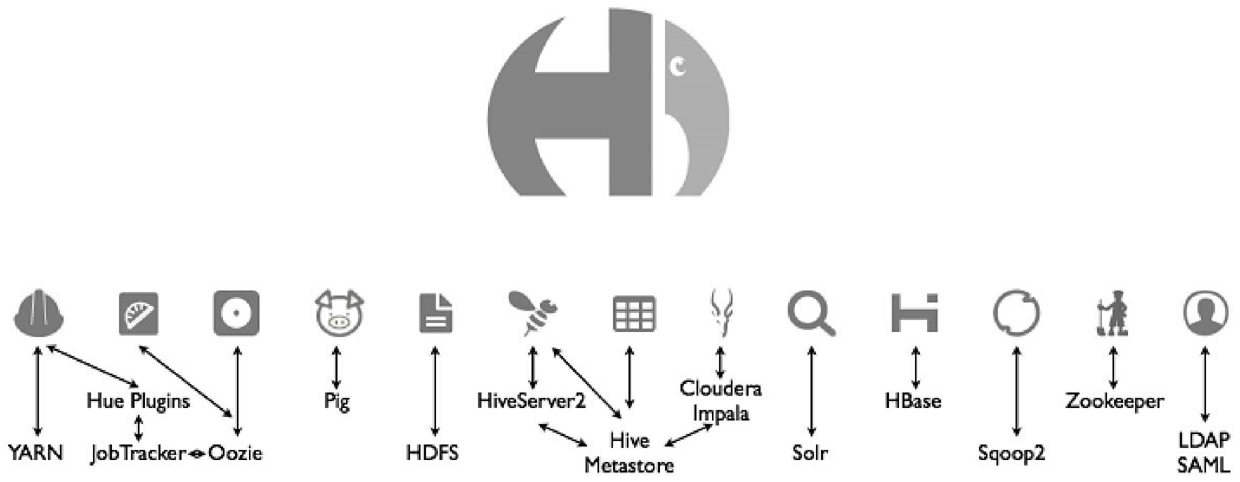
```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie:
hadoop.auth="u=babu&p=babu&t=simple&e=1332977755010&s=JVfT4T785K4jeeLNWXK68rc/0xI=";
Version=1; Path=/
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 28 Mar 2012 13:35:55 GMT

{"Path":"\/user\/babu"}
```

See the WebHDFS REST API web page for complete documentation of the API.

Hue Installation

Hue is a suite of applications that provide web-based access to CDH components and a platform for building custom applications.

The following figure illustrates how Hue works. Hue Server is a "container" web application that sits in between your CDH installation and the browser. It hosts the Hue applications and communicates with various servers that interface with CDH components.



The Hue Server uses a database to manage session, authentication, and Hue application data. For example, the Job Designer application stores job designs in the database.

In a CDH cluster, the Hue Server runs on a special node. For optimal performance, this should be one of the nodes within your cluster, though it can be a remote node as long as there are no overly restrictive firewalls. For small clusters of less than 10 nodes, you can use your existing master node as the Hue Server. In a pseudo-distributed installation, the Hue Server runs on the same machine as the rest of your CDH services.

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera yum, zypper/YaST or apt repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation and the instructions for upgrading to CDH 5 .

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Follow the instructions in the following sections to upgrade, install, configure, and administer Hue.

- Supported Browsers
- Upgrading Hue on page 264
- Installing Hue on page 265
- Configuring CDH Components for Hue on page 267

- Hue Configuration on page 272
- Administering Hue on page 281
- Hue User Guide

Supported Browsers

The Hue UI is supported on the following browsers:

- Windows: Chrome, Firefox 17+, Internet Explorer 9+, Safari 5+
- Linux: Chrome, Firefox 17+
- Mac: Chrome, Firefox 17+, Safari 5+

Upgrading Hue

Note:

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

Upgrading Hue from CDH 4 to CDH 5

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera yum, zypper/YaST or apt repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation and the instructions for upgrading to CDH 5 .

If you have already removed Hue as part of your upgrade to CDH 5, skip to Installing and Configuring Hue.

Step 1: Stop the Hue Server

See Starting and Stopping the Hue Server on page 281.

Step 2: Uninstall the Old Version of Hue

- On RHEL systems:

```
$ sudo yum remove hue-common
```

- On SLES systems:

```
$ sudo zypper remove hue-common
```

- On Ubuntu or Debian systems:

```
sudo apt-get remove hue-common
```

Step 3: Install Hue 3.x

Follow the instructions under Installing Hue.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

Step 4: Start the Hue Server

See Starting and Stopping the Hue Server on page 281.

Upgrading Hue from an Earlier CDH 5 Release

You can upgrade Hue either as part of an overall upgrade to the latest CDH 5 release (see Upgrading from a CDH 5 Beta Release to the Latest Version on page 76) or independently. To upgrade Hue from an earlier CDH 5 release to the latest CDH 5 release, proceed as follows.

Step 1: Stop the Hue Server

See Starting and Stopping the Hue Server on page 281.

Warning:

You **must** stop Hue. If Hue is running during the upgrade, the new version will not work correctly.

Step 2: Install the New Version of Hue

Follow the instructions under Installing Hue on page 265.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

Step 3: Start the Hue Server

See Starting and Stopping the Hue Server on page 281.

Installing Hue

This section describes Hue installation and configuration on a cluster. The steps in this section apply whether you are installing on a single machine in pseudo-distributed mode, or on a cluster.

Hue Installation

Install Python 2.6 or 2.7

Python 2.6 or 2.7 is required to run Hue. RHEL 5 and CentOS 5, in particular, require the EPEL repository package.

In order to install packages from the EPEL repository, first download the appropriate repository rpm packages to your machine and then install Python using yum. For example, use the following commands for RHEL 5 or CentOS 5:

```
$ su -c 'rpm -Uvh
http://download.fedoraproject.org/pub/epel/5/i386/epel-release-5-4.noarch.rpm'
...
$ yum install python26
```

Installing the Hue Packages

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera yum, zypper/YaST or apt repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation and the instructions for upgrading to CDH 5 .

You must install the hue-common package on the machine where you will run the Hue Server. In addition, if you will be using Hue with MRv1, you must install the hue-plugins package on the system where you are running the JobTracker. (In pseudo-distributed mode, these will all be the same system.)

The hue meta-package installs the hue-common package and all the Hue applications; you also need to install hue-server, which contains the Hue start and stop scripts.

Note: If you do not know which system your JobTracker is on, install the hue-plugins package on every node in the cluster.

On RHEL systems:

- On the Hue Server machine, install the hue package:

```
$ sudo yum install hue
```

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the hue-plugins package:

```
$ sudo yum install hue-plugins
```

On SLES systems:

- On the Hue Server machine, install the hue package:

```
$ sudo zypper install hue
```

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the hue-plugins package:

```
$ sudo zypper install hue-plugins
```

On Ubuntu or Debian systems:

- On the Hue Server machine, install the hue package:

```
$ sudo apt-get install hue
```

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the `hue-plugins` package:

```
$ sudo apt-get install hue-plugins
```

> **Important:** For all operating systems, restart the Hue service once installation is complete. See Starting and Stopping the Hue Server on page 281.

Hue Dependencies

The following table shows the components that are dependencies for the different Hue applications and provides links to the installation guides for the required components that are not installed by default.

Component	Required	Applications
HDFS	Yes	Core, File Browser
MapReduce	No	Job Browser, Job Designer, Oozie, Hive Editor, Pig, Sqoop
YARN	No	Job Browser, Job Designer, Oozie, Hive Editor, Pig, Sqoop
Oozie	Yes	Job Designer, Oozie Editor/Dashboard
Hive	Yes	Hive Editor, Metastore Tables
Impala	No	Impala Editor, Metastore Tables
HBase	No	HBase Browser
Pig	No	Pig Editor, Oozie
Search	No	Solr Search
Spark Installation on page 333	No	Spark
Sqoop	No	Oozie
Sqoop 2	No	Sqoop Transfer
ZooKeeper	No	ZooKeeper

Configuring CDH Components for Hue

To enable communication between the Hue Server and CDH components, you must make minor changes to your CDH installation by adding the properties described in this section to your CDH configuration files in `/etc/hadoop-0.20/conf/` or `/etc/hadoop/conf/`. If you are installing on a cluster, make the following configuration changes to your existing CDH installation on **each node** in your cluster.

WebHDFS or HttpFS Configuration

Hue can use either of the following to access HDFS data:

- **WebHDFS** provides high-speed data transfer with good locality because clients talk directly to the DataNodes inside the Hadoop cluster.
- **HttpFS** is a proxy service appropriate for integration with external systems that are not behind the cluster's firewall.

Both WebHDFS and HttpFS use the HTTP REST API so they are fully interoperable, but Hue must be configured to use one or the other. For HDFS HA deployments, you must use HttpFS.

To configure Hue to use either WebHDFS or HttpFS, do the following steps:

1. **For WebHDFS only:**
   a. Add the following property in `hdfs-site.xml` to enable WebHDFS in the NameNode and DataNodes:

   ```
   <property>
     <name>dfs.webhdfs.enabled</name>
     <value>true</value>
   </property>
   ```

   b. Restart your HDFS cluster.
2. Configure Hue as a proxy user for all other users and groups, meaning it may submit a request on behalf of any other user:

   WebHDFS: Add to `core-site.xml`:

   ```
   <!-- Hue WebHDFS proxy user setting -->
   <property>
     <name>hadoop.proxyuser.hue.hosts</name>
     <value>*</value>
   </property>
   <property>
     <name>hadoop.proxyuser.hue.groups</name>
     <value>*</value>
   </property>
   ```

   HttpFS: Verify that `/etc/hadoop-httpfs/conf/httpfs-site.xml` has the following configuration:

   ```
   <!-- Hue HttpFS proxy user setting -->
   <property>
     <name>httpfs.proxyuser.hue.hosts</name>
     <value>*</value>
   </property>
   <property>
     <name>httpfs.proxyuser.hue.groups</name>
     <value>*</value>
   </property>
   ```

   If the configuration is not present, add it to `/etc/hadoop-httpfs/conf/httpfs-site.xml` and restart the HttpFS daemon.
3. Verify that `core-site.xml` has the following configuration:

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

   If the configuration is not present, add it to `/etc/hadoop/conf/core-site.xml` and restart Hadoop.
4. With root privileges, update `hadoop.hdfs_clusters.default.webhdfs_url` in `hue.ini` to point to the address of either WebHDFS or HttpFS.

   ```
   [hadoop]
   [[hdfs_clusters]]
   [[[default]]]
   # Use WebHdfs/HttpFs as the communication mechanism.
   ```

WebHDFS:

```
...
webhdfs_url=http://FQDN:50070/webhdfs/v1/
```

HttpFS:

```
...
webhdfs_url=http://FQDN:14000/webhdfs/v1/
```

Note: If the `webhdfs_url` is uncommented and explicitly set to the empty value, Hue falls back to using the Thrift plugin used in Hue 1.x. This is not recommended.

MRv1 Configuration

Hue communicates with the JobTracker via the Hue plugin, which is a `.jar` file that should be placed in your MapReduce `lib` directory.

Important: The `hue-plugins` package installs the Hue plugins in your MapReduce `lib` directory, `/usr/lib/hadoop/lib`. If you are not using the package-based installation procedure, perform the following steps to install the Hue plugins.

If your JobTracker and Hue Server are located on the same host, copy the file over. If you are currently using CDH 4, your MapReduce library directory might be in `/usr/lib/hadoop/lib`.

```
$ cd /usr/lib/hue
$ cp desktop/libs/hadoop/java-lib/hue-plugins-*.jar /usr/lib/hadoop-0.20-mapreduce/lib
```

If your JobTracker runs on a different host, scp the Hue plugins `.jar` file to the JobTracker host.

Add the following properties to `mapred-site.xml`:

```
<property>
  <name>jobtracker.thrift.address</name>
  <value>0.0.0.0:9290</value>
</property>
<property>
  <name>mapred.jobtracker.plugins</name>
  <value>org.apache.hadoop.thriftfs.ThriftJobTrackerPlugin</value>
  <description>Comma-separated list of jobtracker plug-ins to be activated.</description>
</property>
```

You can confirm that the plugins are running correctly by tailing the daemon logs:

```
$ tail --lines=500 /var/log/hadoop-0.20-mapreduce/hadoop*jobtracker*.log | grep
ThriftPlugin
2009-09-28 16:30:44,337 INFO org.apache.hadoop.thriftfs.ThriftPluginServer: Starting
Thrift server
2009-09-28 16:30:44,419 INFO org.apache.hadoop.thriftfs.ThriftPluginServer:
Thrift server listening on 0.0.0.0:9290
```

Note: If you enable ACLs in the JobTracker, you must add users to the JobTracker `mapred.queue.default.acl-administer-jobs` property in order to allow Hue to display jobs in the Job Browser application. For example, to give the `hue` user access to the JobTracker, you would add the following property:

```
<property>
  <name>mapred.queue.default.acl-administer-jobs</name>
  <value>hue</value>
</property>
```

Repeat this for every user that requires access to the job details displayed by the JobTracker.

If you have any mapred queues besides "default", you must add a property for each queue:

```
<property>
<name>mapred.queue.default.acl-administer-jobs</name>
<value>hue</value>
</property>
<property>
<name>mapred.queue.queue1.acl-administer-jobs</name>
<value>hue</value>
</property>
<property>
<name>mapred.queue.queue2.acl-administer-jobs</name>
<value>hue</value>
</property>
```

Oozie Configuration

In order to run DistCp, Streaming, Pig, Sqoop, and Hive jobs in Job Designer or the Oozie Editor/Dashboard application, you must make sure the Oozie shared libraries are installed for the correct version of MapReduce (MRv1 or YARN). See Installing the Oozie ShareLib in Hadoop HDFS for instructions.

To configure Hue as a default proxy user, add the following properties to `/etc/oozie/conf/oozie-site.xml`:

```
<!-- Default proxyuser configuration for Hue -->
<property>
    <name>oozie.service.ProxyUserService.proxyuser.hue.hosts</name>
    <value>*</value>
</property>
<property>
    <name>oozie.service.ProxyUserService.proxyuser.hue.groups</name>
    <value>*</value>
</property>
```

Search Configuration

See Search Configuration on page 276 for details on how to configure the Search application for Hue.

HBase Configuration

See HBase Configuration on page 276 for details on how to configure the HBase Browser application.

Note: HBase Browser requires Thrift Server 1 to be running.

Hive Configuration

The Beeswax daemon has been replaced by HiveServer2. Hue should therefore point to a running HiveServer2. This change involved the following major updates to the `[beeswax]` section of the Hue configuration file, `hue.ini`.

```
[beeswax]
  # Host where Hive server Thrift daemon is running.
  # If Kerberos security is enabled, use fully-qualified domain name (FQDN).
  ## hive_server_host=<FQDN of HiveServer2>

  # Port where HiveServer2 Thrift server runs on.
  ## hive_server_port=10000
```

Existing Hive Installation

In the Hue configuration file `hue.ini`, modify `hive_conf_dir` to point to the directory containing `hive-site.xml`.

No Existing Hive Installation

Familiarize yourself with the configuration options in `hive-site.xml`. See Hive Installation. Having a `hive-site.xml` is optional but often useful, particularly on setting up a metastore. You can locate it using the `hive_conf_dir` configuration variable.

Permissions

See File System Permissions in the Hive Installation section.

Other Hadoop Settings

HADOOP_CLASSPATH

If you are setting `$HADOOP_CLASSPATH` in your `hadoop-env.sh`, be sure to set it in such a way that user-specified options are preserved. For example:

Correct:

```
# HADOOP_CLASSPATH=<your_additions>:$HADOOP_CLASSPATH
```

Incorrect:

```
# HADOOP_CLASSPATH=<your_additions>
```

This enables certain components of Hue to add to Hadoop's classpath using the environment variable.

hadoop.tmp.dir

If your users are likely to be submitting jobs both using Hue and from the same machine via the command line interface, they will be doing so as the hue user when they are using Hue and via their own user account when they are using the command line. This leads to some contention on the directory specified by `hadoop.tmp.dir`, which defaults to `/tmp/hadoop-${user.name}`. Specifically, `hadoop.tmp.dir` is used to unpack JARs in `/usr/lib/hadoop`. One work around to this is to set `hadoop.tmp.dir` to `/tmp/hadoop-${user.name}-${hue.suffix}` in the `core-site.xml` file:

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/tmp/hadoop-${user.name}-${hue.suffix}</value>
</property>
```

Unfortunately, when the `hue.suffix` variable is unset, you'll end up with directories named `/tmp/hadoop-user.name-${hue.suffix}` in `/tmp`. Despite that, Hue will still work.

Hue Configuration

This section describes configuration you perform in the Hue configuration file `hue.ini`. The location of the Hue configuration file varies depending on how Hue is installed. The location of the Hue configuration folder is displayed when you view the Hue configuration.

- **Note:** Only the root user can edit the Hue configuration file.

Viewing the Hue Configuration

- **Note:** You must be a Hue superuser to view the Hue configuration.

When you log in to Hue, the start-up page displays information about any misconfiguration detected.

To view the Hue configuration, do one of the following:

- Visit `http://`*`myserver`*`:`*`port`* and click the **Configuration** tab.
- Visit `http://`*`myserver`*`:`*`port`*`/dump_config`.

Hue Server Configuration

This section describes Hue Server settings.

Specifying the Hue Server HTTP Address

These configuration properties are under the `[desktop]` section in the Hue configuration file.

Hue uses the CherryPy web server. You can use the following options to change the IP address and port that the web server listens on. The default setting is port 8888 on all configured IP addresses.

```
# Webserver listens on this address and port
http_host=0.0.0.0
http_port=8888
```

Specifying the Secret Key

For security, you should specify the secret key that is used for secure hashing in the session store:

1. Open the Hue configuration file.
2. In the `[desktop]` section, set the `secret_key` property to a long series of random characters (30 to 60 characters is recommended). For example,

   ```
   secret_key=qpbdxoewsqlkhztybvfidtvwekftusgdlofbcfghaswuicmqp
   ```

   - **Note:** If you don't specify a secret key, your session cookies will not be secure. Hue will run but it will also display error messages telling you to set the secret key.

Authentication

By default, the first user who logs in to Hue can choose any username and password and automatically becomes an administrator. This user can create other user and administrator accounts. Hue users should correspond to the Linux users who will use Hue; make sure you use the same name as the Linux username.

By default, user information is stored in the Hue database. However, the authentication system is pluggable. You can configure authentication to use an LDAP directory (Active Directory or OpenLDAP) to perform the authentication, or you can import users and groups from an LDAP directory. See Configuring an LDAP Server for User Admin on page 277.

For more information, see the Hue SDK Documentation.

Configuring the Hue Server for SSL

You can optionally configure Hue to serve over HTTPS. As of CDH 5, pyOpenSSL is now part of the Hue build and does not need to be installed manually. To configure SSL, perform the following steps from the root of your Hue installation path:

1. Configure Hue to use your private key by adding the following options to the Hue configuration file:

```
ssl_certificate=/path/to/certificate
ssl_private_key=/path/to/key
```

Note: Hue can only support a private key without a passphrase.

2. On a production system, you should have an appropriate key signed by a well-known Certificate Authority. If you're just testing, you can create a self-signed key using the `openssl` command that may be installed on your system:

```
# Create a key
$ openssl genrsa 1024 > host.key
# Create a self-signed certificate
$ openssl req -new -x509 -nodes -sha1 -key host.key > host.cert
```

Note: Uploading files using the Hue File Browser over HTTPS requires using a proper SSL Certificate. Self-signed certificates don't work.

Authentication Backend Options for Hue

The table below gives a list of authentication backends Hue can be configured with including the recent SAML backend that enables single sign-on authentication. The `backend` configuration property is available in the `[[auth]]` section under `[desktop]`.

`backend`	`django.contrib.auth.backends.ModelBackend`	This is the default authentication backend used by Django.
	`desktop.auth.backend.AllowAllBackend`	This backend does not require a password for users to log in. All users are automatically authenticated and the username is set to what is provided.
	`desktop.auth.backend.AllowFirstUserDjangoBackend`	This is the default Hue backend. It creates the first user that logs in as the super user. After this, it relies on Django and the user manager to authenticate users.
	`desktop.auth.backend.LdapBackend`	Authenticates users against an LDAP service.
	`desktop.auth.backend.PamBackend`	Authenticates users with PAM (pluggable authentication module). The authentication mode depends on the PAM module used.
	`desktop.auth.backend.SpnegoDjangoBackend`	SPNEGO is an authentication mechanism negotiation protocol. Authentication can be

		delegated to an authentication server, such as a Kerberos KDC, depending on the mechanism negotiated.
	desktop.auth.backend.RemoteUserDjangoBackend	Authenticating remote users with the Django backend. See the Django documentation for more details.
	desktop.auth.backend.OAuthBackend	Delegates authentication to a third-party OAuth server.
	libsaml.backend.SAML2Backend	Secure Assertion Markup Language (SAML) single sign-on (SSO) backend. Delegates authentication to the configured Identity Provider. See Configuring Hue for SAML for more details.

Note: All backends that delegate authentication to a third-party authentication server eventually import users into the Hue database. While the metadata is stored in the database, user authentication will still take place outside Hue.

Beeswax Configuration

In the `[beeswax]` section of the configuration file, you can optionally specify the following:

hive_server_host	The fully-qualified domain name or IP address of the host running HiveServer2.
hive_server_port	The port of the HiveServer2 Thrift server. Default: 10000.
hive_conf_dir	The directory containing `hive-site.xml`, the HiveServer2 configuration file.

Cloudera Impala Query UI Configuration

In the `[impala]` section of the configuration file, you can optionally specify the following:

server_host	The hostname or IP address of the Impala Server. Default: localhost.
server_port	The port of the Impalad Server. Default: `21050`
impersonation_enabled	Turn on/off impersonation mechanism when talking to Impala. Default: `False`

DB Query Configuration

The DB Query app can have any number of databases configured in the `[[databases]]` section under `[librdbms]`. A database is known by its section name (`sqlite`, `mysql`, `postgresql`, and `oracle` as in the list below).

<table>
<tr><th>Database Type</th><th>Configuration Properties</th></tr>
<tr><td>SQLite: <code>[[[sqlite]]]</code></td><td><pre>
# Name to show in the UI.
## nice_name=SQLite

# For SQLite, name defines the path to the database.
## name=/tmp/sqlite.db

# Database backend to use.
## engine=sqlite
</pre></td></tr>
<tr><td>MySQL, Oracle or PostgreSQL:

<code>[[[mysql]]]</code>

<b>Note:</b> Replace with <code>oracle</code> or <code>postgresql</code> as required.</td><td><pre>
# Name to show in the UI.
## nice_name="My SQL DB"

# For MySQL and PostgreSQL, name is the name of the
database.
# For Oracle, Name is instance of the Oracle server. For
 express edition
# this is 'xe' by default.
## name=mysqldb

# Database backend to use. This can be:
# 1. mysql
# 2. postgresql
# 3. oracle
## engine=mysql

# IP or hostname of the database to connect to.
## host=localhost

# Port the database server is listening to. Defaults are:
# 1. MySQL: 3306
# 2. PostgreSQL: 5432
# 3. Oracle Express Edition: 1521
## port=3306

# Username to authenticate with when connecting to the
database.
## user=example

# Password matching the username to authenticate with
when
# connecting to the database.
## password=example
</pre></td></tr>
</table>

Pig Editor Configuration

In the `[pig]` section of the configuration file, you can optionally specify the following:

`remote_data_dir`	Location on HDFS where the Pig examples are stored.

Sqoop Configuration

In the `[sqoop]` section of the configuration file, you can optionally specify the following:

`server_url`	The URL of the sqoop2 server.

Job Browser Configuration

By default, any user can see submitted job information for all users. You can restrict viewing of submitted job information by optionally setting the following property under the `[jobbrowser]` section in the Hue configuration file:

`share_jobs`	Indicate that jobs should be shared with all users. If set to false, they will be visible only to the owner and administrators.

Job Designer

In the `[jobsub]` section of the configuration file, you can optionally specify the following:

`remote_data_dir`	Location in HDFS where the Job Designer examples and templates are stored.

Oozie Editor/Dashboard Configuration

By default, any user can see all workflows, coordinators, and bundles. You can restrict viewing of workflows, coordinators, and bundles by optionally specifying the following property under the `[oozie]` section of the Hue configuration file:

`share_jobs`	Indicate that workflows, coordinators, and bundles should be shared with all users. If set to false, they will be visible only to the owner and administrators.
`oozie_jobs_count`	Maximum number of Oozie workflows or coordinators or bundles to retrieve in one API call.
`remote_data_dir`	The location in HDFS where Oozie workflows are stored.

Also see Liboozie Configuration on page 280

Search Configuration

In the `[search]` section of the configuration file, you can optionally specify the following:

`security_enabled`	Indicate whether Solr requires clients to perform Kerberos authentication.
`empty_query`	Query sent when no term is entered. Default: `*:*`.
`solr_url`	URL of the Solr server.

HBase Configuration

In the `[hbase]` section of the configuration file, you can optionally specify the following:

`truncate_limit`	Hard limit of rows or columns per row fetched before truncating. Default: 500
`hbase_clusters`	Comma-separated list of HBase Thrift servers for clusters in the format of "(name|host:port)". Default: `(Cluster|localhost:9090)`

User Admin Configuration

In the `[useradmin]` section of the configuration file, you can optionally specify the following:

`default_user_group`	The name of the group to which a manually created user is automatically assigned.

	Default: default.

Configuring an LDAP Server for User Admin

User Admin can interact with an LDAP server, such as Active Directory, in one of two ways:

- You can import user and group information from your current Active Directory infrastructure using the LDAP Import feature in the User Admin application. User authentication is then performed by User Admin based on the imported user and password information. You can then manage the imported users, along with any users you create directly in User Admin. See Enabling Import of Users and Groups from an LDAP Directory on page 277.
- You can configure User Admin to use an LDAP server as the authentication back end, which means users logging in to Hue will authenticate to the LDAP server, rather than against a username and password kept in User Admin. In this scenario, your users must all reside in the LDAP directory. See Enabling the LDAP Server for User Authentication on page 278 for further information.

Enabling Import of Users and Groups from an LDAP Directory

User Admin can import users and groups from an Active Directory via the Lightweight Directory Authentication Protocol (LDAP). In order to use this feature, you must configure User Admin with a set of LDAP settings in the Hue configuration file.

- **Note:** If you import users from LDAP, you must set passwords for them manually; password information is not imported.

To enable LDAP import of users and groups:

1. In the Hue configuration file, configure the following properties in the `[[ldap]]` section:

Property	Description	Example
`base_dn`	The search base for finding users and groups.	`base_dn="DC=mycompany,DC=com"`
`nt_domain`	The NT domain to connect to (only for use with Active Directory).	`nt_domain=mycompany.com`
`ldap_url`	URL of the LDAP server.	`ldap_url=ldap://auth.mycompany.com`
`ldap_cert`	Path to certificate for authentication over TLS (optional).	`ldap_cert=/mycertsdir/myTLScert`
`bind_dn`	Distinguished name of the user to bind as – not necessary if the LDAP server supports anonymous searches.	`bind_dn="CN=ServiceAccount,DC=mycompany,DC=com"`
`bind_password`	Password of the bind user – not necessary if the LDAP server supports anonymous searches.	`bind_password=P@ssw0rd`

2. Configure the following properties in the `[[[users]]]` section:

Property	Description	Example
`user_filter`	Base filter for searching for users.	`user_filter="objectclass=*"`
`user_name_attr`	The username attribute in the LDAP schema.	`user_name_attr=sAMAccountName`

3. Configure the following properties in the `[[[groups]]]` section:

Property	Description	Example
`group_filter`	Base filter for searching for groups.	`group_filter="objectclass=*"`
`group_name_attr`	The username attribute in the LDAP schema.	`group_name_attr=cn`

> **Note:** If you provide a TLS certificate, it must be signed by a Certificate Authority that is trusted by the LDAP server.

Enabling the LDAP Server for User Authentication

You can configure User Admin to use an LDAP server as the authentication back end, which means users logging in to Hue will authenticate to the LDAP server, rather than against usernames and passwords managed by User Admin.

> **Important:** Be aware that when you enable the LDAP back end for user authentication, user authentication by User Admin will be disabled. This means there will be no superuser accounts to log into Hue unless you take one of the following actions:
>
> - Import one or more superuser accounts from Active Directory and assign them superuser permission.
> - If you have already enabled the LDAP authentication back end, log into Hue using the LDAP back end, which will create a LDAP user. Then disable the LDAP authentication back end and use User Admin to give the superuser permission to the new LDAP user.

After assigning the superuser permission, enable the LDAP authentication back end.

To enable the LDAP server for user authentication:

1. In the Hue configuration file, configure the following properties in the `[[ldap]]` section:

Property	Description	Example
`ldap_url`	URL of the LDAP server, prefixed by `ldap://` or `ldaps://`	`ldap_url=ldap://auth.mycompany.com`
`search_bind_authentication`	Search bind authentication is now the default instead of direct bind. To revert to direct bind, the value of this property should be set to `false`. When using search bind semantics, Hue will ignore the following `nt_domain` and `ldap_username_pattern` properties.	`search_bind_authentication=false`
`nt_domain`	The NT domain over which the user connects (not strictly necessary if using `ldap_username_pattern`.	`nt_domain=mycompany.com`
`ldap_username_pattern`	Pattern for searching for usernames – Use `<username>` for the username parameter. For use when using `LdapBackend` for Hue authentication	`ldap_username_pattern="uid=<username>,ou=People,dc=mycompany,dc=com"`

2. If you are using TLS or secure ports, add the following property to specify the path to a TLS certificate file:

Property	Description	Example
`ldap_cert`	Path to certificate for authentication over TLS. **Note:** If you provide a TLS certificate, it must be signed by a Certificate Authority that is trusted by the LDAP server.	`ldap_cert=/mycertsdir/myTLScert`

3. In the`[[auth]]` sub-section inside `[desktop]` change the following:

`backend`	Change the setting of backend from `backend=desktop.auth.backend.AllowFirstUserDjangoBackend` to `backend=desktop.auth.backend.LdapBackend`

Hadoop Configuration

The following configuration variables are under the `[hadoop]` section in the Hue configuration file.

HDFS Cluster Configuration

Hue currently supports only one HDFS cluster, which you define under the `[[hdfs_clusters]]` sub-section. The following properties are supported:

`[[[default]]]`	The section containing the default settings.
`fs_defaultfs`	The equivalent of `fs.defaultFS` (also referred to as `fs.default.name`) in a Hadoop configuration.
`webhdfs_url`	The HttpFS URL. The default value is the HTTP port on the NameNode.

YARN (MRv2) and MapReduce (MRv1) Cluster Configuration

Job Browser can display both MRv1 and MRv2 jobs, but must be configured to display one type at a time by specifying either `[[yarn_clusters]]` or `[[mapred_clusters]]` sections in the Hue configuration file.

The following YARN cluster properties are defined under the under the `[[yarn_clusters]]` sub-section:

`[[[default]]]`	The section containing the default settings.
`resourcemanager_host`	The fully-qualified domain name of the host running the ResourceManager.
`resourcemanager_port`	The port for the ResourceManager IPC service.
`submit_to`	If your Oozie is configured to use a YARN cluster, then set this to true. Indicate that Hue should submit jobs to this YARN cluster.
`proxy_api_url`	URL of the ProxyServer API. Default: `http://localhost:8088`
`history_server_api_url`	URL of the HistoryServer API Default: `http://localhost:19888`

The following MapReduce cluster properties are defined under the `[[mapred_clusters]]` sub-section:

`[[[default]]]`	The section containing the default settings.
`jobtracker_host`	The fully-qualified domain name of the host running the JobTracker.
`jobtracker_port`	The port for the JobTracker IPC service.
`submit_to`	If your Oozie is configured with to use a 0.20 MapReduce service, then set this to true. Indicate that Hue should submit jobs to this MapReduce cluster.

Note: High Availability (MRv1):

Add High Availability (HA) support for your MRv1 cluster by specifying a failover JobTracker. You can do this by configuring the following property under the `[[[ha]]]` sub-section for MRv1.

```
# Enter the host on which you are running the failover JobTracker
# jobtracker_host=<localhost-ha>
```

High Availability (YARN):

Add the following `[[[ha]]]` section under the `[hadoop] > [[yarn_clusters]]` sub-section in `hue.ini` with configuration properties for a second ResourceManager. As long as you have the `logical_name` property specified as below, jobs submitted to Oozie will work. The Job Browser, however, will *not* work with HA in this case.

```
[[[ha]]]
resourcemanager_host=<second_resource_manager_host_FQDN>
resourcemanager_api_url=http://<second_resource_manager_host_URL>
proxy_api_url=<second_resource_manager_proxy_URL>
history_server_api_url=<history_server_API_URL>
resourcemanager_port=<port_for_RM_IPC>
security_enabled=false
submit_to=true
logical_name=XXXX
```

Liboozie Configuration

In the `[liboozie]` section of the configuration file, you can optionally specify the following:

`security_enabled`	Indicate whether Oozie requires clients to perform Kerberos authentication.
`remote_deployement_dir`	The location in HDFS where the workflows and coordinators are deployed when submitted by a non-owner.
`oozie_url`	The URL of the Oozie server.

ZooKeeper Configuration

In the `[zookeeper]` section of the configuration file, you can optionally specify the following:

`host_ports`	Comma-separated list of ZooKeeper servers in the format "host:port". Example: `localhost:2181,localhost:2182,localhost:2183`
`rest_url`	The URL of the REST Contrib service (required for znode browsing). Default: `http://localhost:9998`

Setting up REST Service for ZooKeeper

ZooKeeper Browser requires the ZooKeeper REST service to be running. Follow the instructions below to set this up.

Step 1: Git and build the ZooKeeper repository

```
git clone https://github.com/apache/zookeeper
cd zookeeper
ant
Buildfile: /home/hue/Development/zookeeper/build.xml

init:
[mkdir] Created dir: /home/hue/Development/zookeeper/build/classes
[mkdir] Created dir: /home/hue/Development/zookeeper/build/lib
[mkdir] Created dir: /home/hue/Development/zookeeper/build/package/lib
[mkdir] Created dir: /home/hue/Development/zookeeper/build/test/lib
…
```

Step 2: Start the REST service

```
cd src/contrib/rest
nohup ant run&
```

Step 3: Update ZooKeeper configuration properties (if required)

If ZooKeeper and the REST service are not on the same machine as Hue, update the Hue configuration file and specify the correct hostnames and ports as shown in the sample configuration below:

```
[zookeeper]
 ...
 [[clusters]]
  ...
  [[[default]]]
          # Zookeeper ensemble. Comma separated list of Host/Port.
          # e.g. localhost:2181,localhost:2182,localhost:2183
          ## host_ports=localhost:2181

          # The URL of the REST contrib service
          ## rest_url=http://localhost:9998
```

You should now be able to successfully run the ZooKeeper Browser app.

Administering Hue

The following sections contain details about managing and operating a Hue installation:

- Starting and Stopping the Hue Server on page 281
- Configuring Your Firewall for Hue on page 282
- Anonymous Usage Data Collection on page 282
- Managing Hue Processes on page 282
- Viewing Hue Logs on page 282

Starting and Stopping the Hue Server

The `hue-server` package includes service scripts to start and stop the Hue Server.

To start the Hue Server:

```
$ sudo service hue start
```

To restart the Hue Server:

```
$ sudo service hue restart
```

To stop the Hue Server:

```
$ sudo service hue stop
```

Configuring Your Firewall for Hue

Hue currently requires that the machines within your cluster can connect to each other freely over TCP. The machines outside your cluster must be able to open TCP port 8888 on the Hue Server (or the configured Hue web HTTP port) to interact with the system.

Anonymous Usage Data Collection

Hue tracks anonymised pages and application versions in order to gather information to help compare each application's usage levels. The data collected does not include any hostnames or IDs. For example, the data is of the form: `/2.3.0/pig`, `/2.5.0/beeswax/execute`.

For Hue 2.5.0 (and later), you can restrict this data collection by setting the `collect_usage` property to `false` under the `[desktop]` section in the Hue configuration file, `hue.ini`.

```
[desktop]
...
# Help improve Hue with anonymous usage analytics.
# Use Google Analytics to see how many times an application or specific section of an
 application is used, nothing more.
## collect_usage=false
```

If you are using an earlier version of Hue, disable this data collection by navigating to Step 3 of Hue's Quick Start Wizard. Under **Anonymous usage analytics**, uncheck the **Check to enable usage analytics** checkbox.

Managing Hue Processes

A script called `supervisor` manages all Hue processes. The supervisor is a watchdog process; its only purpose is to spawn and monitor other processes. A standard Hue installation starts and monitors the `runcpserver` process.

- `runcpserver` – a web server that provides the core web functionality of Hue

If you have installed other applications into your Hue instance, you may see other daemons running under the supervisor as well.

You can see the supervised processes running in the output of `ps -f -u hue`.

Note that the supervisor automatically restarts these processes if they fail for any reason. If the processes fail repeatedly within a short time, the supervisor itself shuts down.

Viewing Hue Logs

You can view the Hue logs in the `/var/log/hue` directory, where you can find:

- An `access.log` file, which contains a log for all requests against the Hue Server.
- A `supervisor.log` file, which contains log information for the supervisor process.
- A `supervisor.out` file, which contains the stdout and stderr for the supervisor process.
- A `.log` file for each supervised process described above, which contains the logs for that process.
- A `.out` file for each supervised process described above, which contains the stdout and stderr for that process.

If users on your cluster have problems running Hue, you can often find error messages in these log files.

Viewing Recent Log Messages

In addition to logging `INFO` level messages to the logs directory, the Hue Server keeps a small buffer of log messages at all levels in memory. The `DEBUG` level messages can sometimes be helpful in troubleshooting issues.

In the Hue UI you can view these messages by selecting the **Server Logs** tab in the **About** application. You can also view these logs by visiting `http://`*`myserver`*`:`*`port`*`/logs`.

Hue Database

The Hue server requires an SQL database to store small amounts of data, including user account information as well as history of job submissions and Hive queries. The Hue server supports a lightweight embedded database and several types of external databases. If you elect to configure Hue to use an external database, upgrades may require more manual steps.

Embedded Database

By default, Hue is configured to use the embedded database SQLite for this purpose, and should require no configuration or management by the administrator.

Inspecting the Embedded Hue Database

The default SQLite database used by Hue is located in `/var/lib/hue/desktop.db`. You can inspect this database from the command line using the `sqlite3` program. For example:

```
# sqlite3 /var/lib/hue/desktop.db
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> select username from auth_user;
admin
test
sample
sqlite>
```

Important: It is strongly recommended that you avoid making any modifications to the database directly using `sqlite3`, though `sqlite3` is useful for management or troubleshooting.

Backing up the Embedded Hue Database

If you use the default embedded SQLite database, copy the `desktop.db` file to another node for backup. It is recommended that you back it up on a regular schedule, and also that you back it up before any upgrade to a new version of Hue.

External Database

Although SQLite is the default database, some advanced users may prefer to have Hue access an external database. Hue supports MySQL, PostgreSQL, and Oracle. See Supported Databases for the supported versions.

Note: In the instructions that follow, dumping the database and editing the JSON objects is only necessary if you have data in SQLite that you need to migrate. If you don't need to migrate data from SQLite, you can skip those steps.

Configuring the Hue Server to Store Data in MySQL

1. Shut down the Hue server if it is running.

2. Dump the existing database data to a text file. Note that using the `.json` extension is required.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field.
4. Start the Hue server.
5. Install the MySQL client developer package.

OS	Command
RHEL	`$ sudo yum install mysql-devel`
SLES	`$ sudo zypper install mysql-devel`
Ubuntu or Debian	`$ sudo apt-get install libmysqlclient-dev`

6. Install the MySQL connector.

OS	Command
RHEL	`$ sudo yum install mysql-connector-java`
SLES	`$ sudo zypper install mysql-connector-java`
Ubuntu or Debian	`$ sudo apt-get install libmysql-java`

7. Install and start MySQL.

OS	Command
RHEL	`$ sudo yum install mysql-server`

SLES	`$ sudo zypper install mysql`
	`$ sudo zypper install libmysqlclient_r15`

Ubuntu or Debian	`$ sudo apt-get install mysql-server`

8. Change the `/etc/my.cnf` file as follows:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
bind-address=<ip-address>
default-storage-engine=MyISAM
```

9. Start the `mysql` daemon.

OS	Command
RHEL	`$ sudo service mysqld start`
SLES and Ubuntu or Debian	`$ sudo service mysql start`

10. Configure MySQL to use a strong password. In the following procedure, your current `root` password is blank. Press the **Enter** key when you're prompted for the root password.

```
$ sudo /usr/bin/mysql_secure_installation
[...]
```

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

11. Configure MySQL to start at boot.

OS	Command

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

SLES	`$ sudo chkconfig --add mysql`
Ubuntu or Debian	`$ sudo chkconfig mysql on`

12. Create the Hue database and grant privileges to a hue user to manage the database.

```
mysql> create database hue;
Query OK, 1 row affected (0.01 sec)
mysql> grant all on hue.* to 'hue'@'localhost' identified by '<secretpassword>';
Query OK, 0 rows affected (0.00 sec)
```

13. Open the Hue configuration file in a text editor.
14. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your MySQL setup):

```
host=localhost
port=3306
engine=mysql
user=hue
password=<secretpassword>
name=hue
```

15. As the hue user, load the existing data and create the necessary database tables using syncdb and `migrate`.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
$ mysql -uhue -p<secretpassword>
mysql > SHOW CREATE TABLE auth_permission;
```

16. (**InnoDB only**) Drop the foreign key.

```
mysql > ALTER TABLE auth_permission DROP FOREIGN KEY content_type_id_refs_id_XXXXXX;
```

17. Delete the rows in the `django_content_type` table.

```
mysql > DELETE FROM hue.django_content_type;
```

18. Load the data.

```
$ <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

19. **(InnoDB only)** Add the foreign key.

```
$ mysql -uhue -p<secretpassword>
mysql > ALTER TABLE auth_permission ADD FOREIGN KEY (`content_type_id`) REFERENCES
 `django_content_type` (`id`);
```

Configuring the Hue Server to Store Data in PostgreSQL

Warning: Hue requires PostgreSQL 8.4 or newer.

1. Shut down the Hue server if it is running.
2. Dump the existing database data to a text file. Note that using the `.json` extension is required.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field.
4. Install required packages.

OS	Command
RHEL	`$ sudo yum install postgresql-devel gcc python-devel`
SLES	`$ sudo zypper install postgresql-devel gcc python-devel`
Ubuntu or Debian	`$ sudo apt-get install postgresql-devel gcc python-devel`

5. Install the module that provides the connector to PostgreSQL.

```
sudo -u hue <HUE_HOME>/build/env/bin/pip install setuptools
sudo -u hue <HUE_HOME>/build/env/bin/pip install psycopg2
```

6. Install the PostgreSQL server.

OS	Command
RHEL	`$ sudo yum install postgresql-server`
SLES	`$ sudo zypper install postgresql-server`
Ubuntu or Debian	`$ sudo apt-get install postgresql`

7. Initialize the data directories:

```
$ service postgresql initdb
```

8. Configure client authentication.

   a. Edit `/var/lib/pgsql/data/pg_hba.conf`.
   b. Set the authentication methods for local to `trust` and for host to `password` and add the following line at the end.

   ```
   host hue hue 0.0.0.0/0 md5
   ```

9. Start the PostgreSQL server.

```
$ su - postgres
# /usr/bin/postgres -D /var/lib/pgsql/data > logfile 2>&1 &
```

10. Configure PostgreSQL to listen on all network interfaces.

    Edit `/var/lib/pgsql/data/postgresql.conf` and set list_addresses:

```
listen_addresses = ‘0.0.0.0’     # Listen on all addresses
```

11. Create the hue database and grant privileges to a hue user to manage the database.

```
# psql -U postgres
postgres=# create database hue;
postgres=# \c hue;
You are now connected to database 'hue'.
postgres=# create user hue with password '<secretpassword>';
postgres=# grant all privileges on database hue to hue;
postgres=# \q
```

12. Restart the PostgreSQL server.

```
$ sudo service postgresql restart
```

13. Verify connectivity.

```
psql –h localhost –U hue –d hue
Password for user hue: <secretpassword>
```

14. Configure the PostgreSQL server to start at boot.

OS	Command

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

SLES	`$ sudo chkconfig --add postgresql`
Ubuntu or Debian	`$ sudo chkconfig postgresql on`

15. Open the Hue configuration file in a text editor.
16. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your PostgreSQL setup).

```
host=localhost
port=5432
engine=postgresql_psycopg2
user=hue
password=<secretpassword>
name=hue
```

17. As the hue user, configure Hue to load the existing data and create the necessary database tables. You will need to run both the `migrate` and `syncdb` commands.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
```

18. Determine the foreign key ID.

```
bash# su – postgres
$ psql –h localhost –U hue –d hue
postgres=# \d auth_permission;
```

19. Drop the foreign key that you retrieved in the previous step.

```
postgres=# ALTER TABLE auth_permission DROP CONSTRAINT
content_type_id_refs_id_<XXXXXX>;
```

20. Delete the rows in the `django_content_type` table.

```
postgres=# TRUNCATE django_content_type CASCADE;
```

21. Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

22. Add back the foreign key you dropped.

```
bash# su – postgres
$ psql –h localhost –U hue –d hue
postgres=# ALTER TABLE auth_permission ADD CONSTRAINT
content_type_id_refs_id_<XXXXXX> FOREIGN KEY (content_type_id) REFERENCES
django_content_type(id) DEFERRABLE INITIALLY DEFERRED;
```

Configuring the Hue Server to Store Data in Oracle

1. Ensure Python 2.6 or newer is installed on the server Hue is running on.
2. Download the Oracle client libraries at Instant Client for Linux x86-64 Version 11.1.0.7.0, Basic and SDK (with headers) zip files to the same directory.
3. Unzip the zip files.
4. Set environment variables to reference the libraries.

```
$ export ORACLE_HOME=<download directory>
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME
```

5. Create a symbolic link for the shared object:

```
$ cd $ORACLE_HOME
$ ln -sf libclntsh.so.11.1 libclntsh.so
```

6. Install the Python Oracle library.

```
$ <HUE_HOME>/build/env/bin/pip install cx_Oracle
```

7. Upgrade django south.

```
$ <HUE_HOME>/build/env/bin/pip install south --upgrade
```

8. Get a data dump by executing:

```
$ <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json --indent 2
```

9. Open the Hue configuration file in a text editor.

10. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your Oracle setup):

```
host=localhost
port=1521
engine=oracle
user=hue
password=<secretpassword>
name=<SID of the Oracle database, for example, 'XE'>
```

To add support for a multithreaded environment, set the `threaded` option to `true` under the `[desktop]>[[database]]` section.

```
options={'threaded':true}
```

11. As the hue user, configure Hue to load the existing data and create the necessary database tables. You will need to run both the `syncdb` and `migrate` commands.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ sudo -u hue <HUE_HOME>/build/env/bin/hue migrate
```

12. Generate statements to delete all data from Oracle tables:

```
> SELECT 'DELETE FROM ' || '.' || table_name || ';' FROM user_tables;
```

13. Run the statements generated in the preceding step.
14. Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

Viewing the Hue User Guide

For additional information about Hue, see the Hue User Guide.

Llama Installation

Llama is a system that mediates resource management between Cloudera Impala and Hadoop YARN. Llama enables Impala to reserve, use, and release resource allocations in a Hadoop cluster. Llama is only required if resource management is enabled in Impala.

Note: At this point Llama has been tested only in a Cloudera Manager deployment. For information on using Cloudera Manager to configure Llama and Impala, see Installing Impala with Cloudera Manager.

See the following sections for information and instructions:

- Prerequisites
- Packaging
- Installing Llama on page 291
- Configuring Llama
- Starting, Stopping, and Using Llama

To configure Llama in CDH 5 with Kerberos security, see Llama Security Configuration.

Prerequisites

- An operating system supported by CDH 5
- Oracle JDK; see also Java Development Kit Installation on page 397

Packaging

The packaging options for installing Llama are:

- A RPM package for the Llama ApplicationMaster(`llama-master`)
- A Debian package for the Llama ApplicationMaster(`llama-master`)

Installing Llama

Llama is distributed in two packages:

- `llama`- the binaries and configuration files
- `llama-master` - the service script that you use to run Llama

Installing the `llama-master` package installs the `llama` package and the dependencies needed to run Llama, creating a `llama` service configured to start Llama at system startup time.

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install Llama. For instructions, see CDH 5 Installation on page 27.

To install Llama on a Red Hat system:

```
$ sudo yum install llama-master
```

To install Llama on an Ubuntu or other Debian system:

```
$ sudo apt-get install llama-master
```

To install Llama on a SLES system:

```
$ sudo zypper install llama-master
```

Configuring Llama

Important:

Llama works with YARN only; it does not work with MRv1.

When you install Llama from an RPM or Debian package, Llama server creates all configuration, documentation, and runtime files in the standard Linux directories, as follows.

Type of File	Where Installed
Binaries	`/usr/lib/llama/`
Configuration	`/etc/llama/conf/`
Logs	`/var/log/llama/`
PID file	`/var/run/llama/`

Llama uses the YARN configuration to interact with Hadoop. The Llama configuration file, `/etc/llama/conf/llama-site.xml`, contains all default values after installation. You do not need to change these to get Llama up and running.

Starting, Stopping, and Using Llama

To start Llama

After you have completed all the required configuration steps, you can start Llama:

```
$ sudo service llama start
```

If you see the message `LlamaAMServer - Llama started!` in the `llama.log` log file, the system has started successfully.

To stop Llama

Stop Llama as follows

```
$ sudo service llama stop
```

Mahout Installation

Apache Mahout is a machine-learning tool. By enabling you to build machine-learning libraries that are scalable to "reasonably large" datasets, it aims to make building intelligent applications easier and faster.

Note:

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

The main use cases for Mahout are:

- **Recommendation mining**, which tries to identify things users will like on the basis of their past behavior (for example shopping or online-content recommendations)
- **Clustering**, which groups similar items (for example, documents on similar topics)
- **Classification**, which learns from existing categories what members of each category have in common, and on that basis tries to categorize new items
- **Frequent item-set mining**, which takes a set of item-groups (such as terms in a query session, or shopping-cart content) and identifies items that usually appear together

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the instructions below to install Mahout. For instructions, see Installing CDH 5 on page 27.

Use the following sections to install, update and use Mahout:

- Upgrading Mahout
- Installing Mahout
- The Mahout Executable
- Getting Started
- The Apache Mahout Wiki

Upgrading Mahout

Note:

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

Upgrading Mahout from CDH 4 to CDH 5

To upgrade Mahout to CDH 5, you must uninstall the CDH 4 version and then install the CDH 5 version. Proceed as follows.

Step 1: Remove CDH 4 Mahout

To remove Mahout on a Red Hat system:

```
$ sudo yum remove mahout
```

To remove Mahout on a SLES system:

```
$ sudo zypper remove mahout
```

To remove Mahout on an Ubuntu or Debian system:

```
$ sudo apt-get remove mahout
```

Step 2: Install CDH 5 Mahout

See Installing Mahout.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Upgrading Mahout from an Earlier CDH 5 Release to the Latest CDH 5 Release

To upgrade Mahout to the latest release, simply install the new version; see Installing Mahout on page 294.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Installing Mahout

You can install Mahout from an RPM or Debian package, or from a tarball.

Note:

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

Installing from packages is more convenient than installing the tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations

These instructions assume that you will install from packages if possible.

- **Important:**

  If you have not already done so, install Cloudera's yum, zypper/YaST or apt repository before using the instructions below to install Mahout. For instructions, see Installing CDH 5 on page 27.

To install Mahout on a Red Hat system:

```
$ sudo yum install mahout
```

To install Mahout on a SLES system:

```
$ sudo zypper install mahout
```

To install Mahout on an Ubuntu or Debian system:

```
$ sudo apt-get install mahout
```

To access Mahout documentation:

The Mahout docs are bundled in a mahout-doc package that should be installed separately.

```
$ sudo apt-get install mahout-doc
```

The contents of this package are saved under /usr/share/doc/mahout*.

The Mahout Executable

The Mahout executable is installed in /usr/bin/mahout. Use this executable to run your analysis.

Getting Started with Mahout

To get started with Mahout, you can follow the instructions in this Apache Mahout Quickstart.

The Mahout Documentation

For more information about Mahout, see the Apache Mahout Wiki.

Oozie Installation

About Oozie

Apache Oozie Workflow Scheduler for Hadoop is a workflow and coordination service for managing Apache Hadoop jobs:

- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of *actions*; *actions* are typically Hadoop jobs (MapReduce, Streaming, Pipes, Pig, Hive, Sqoop, etc).
- Oozie Coordinator jobs trigger recurrent Workflow jobs based on time (frequency) and data availability.
- Oozie Bundle jobs are sets of Coordinator jobs managed as a single job.

Oozie is an extensible, scalable and data-aware service that you can use to orchestrate dependencies among jobs running on Hadoop.

- To find out more about Oozie, see http://archive.cloudera.com/cdh5/cdh/5/oozie/.
- To install or upgrade Oozie, follow the directions on this page.

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Oozie Packaging

There are two packaging options for installing Oozie:

- Separate RPM packages for the Oozie server (`oozie`) and client (`oozie-client`)
- Separate Debian packages for the Oozie server (`oozie`) and client (`oozie-client`)

You can also download an Oozie tarball.

Oozie Prerequisites

- Prerequisites for installing Oozie server:

  - An operating system supported by CDH 5
  - Oracle JDK
  - A supported database if you are not planning to use the default (Derby).
- Prerequisites for installing Oozie client:
  - Oracle JDK

Note:

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

Upgrading Oozie

Follow these instructions to upgrade Oozie to CDH 5 from RPM or Debian Packages.

Important:

After upgrading Oozie, but before restarting Oozie, check for any custom JARs you may have created in `/var/lib/oozie` and make sure you recompile all classes in these JARs with the new version of Oozie and its dependencies. If you don't do this, Oozie may fail to start, or behave in unexpected ways.

Upgrading Oozie from CDH 4 to CDH 5

Before you start:

Make sure there are no workflows in `RUNNING` or `SUSPENDED` status; otherwise the database upgrade will fail and you will have to reinstall Oozie CDH 4 to complete or kill those running workflows.

To upgrade Oozie from CDH 4 to CDH 5, back up the configuration files and database, uninstall the CDH 4 version and then install and configure the CDH 5 version. Proceed as follows.

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5, you can skip Step 1 below and proceed with installing the new CDH 5 version of Oozie.

Important: Ubuntu and Debian upgrades

When you uninstall CDH 4 Oozie on Ubuntu and Debian systems, the contents of `/var/lib/oozie` are removed, leaving a bare directory. This can cause the Oozie upgrade to CDH 5 to fail. To prevent this, either copy the database files to another location and restore them after the uninstall, or recreate them after the uninstall. Make sure you do this before starting the re-install.

Step 1: Remove Oozie

1. Back up the Oozie configuration files in `/etc/oozie` and the Oozie database. For convenience you may want to save Oozie configuration files in your home directory; you will need them after installing the new version of Oozie.
2. Stop the Oozie Server.

To stop the Oozie Server:

```
sudo service oozie stop
```

3. Uninstall Oozie.

   To uninstall Oozie, run the appropriate command on each host:

   - On Red Hat-compatible systems:

     ```
     $ sudo yum remove oozie
     ```

     ```
     $ sudo yum remove oozie-client
     ```

   - On SLES systems:

     ```
     $ sudo zypper remove  oozie
     ```

     ```
     $ sudo zypper remove oozie-client
     ```

   - On Ubuntu or Debian systems:

     ```
     sudo apt-get remove oozie
     ```

     ```
     sudo apt-get remove oozie-client
     ```

Step 2: Install Oozie

Follow the procedure under Installing Oozie and then proceed to Configuring Oozie after Upgrading from CDH 4 on page 302. For packaging information, see Oozie Packaging.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Upgrading Oozie from an Earlier CDH 5 Release

Before you start:

Make sure there are no workflows in `RUNNING` or `SUSPENDED` status; otherwise the database upgrade will fail and you will have to reinstall Oozie CDH 4 to complete or kill those running workflows.

The steps that follow assume you are upgrading Oozie as part of an overall upgrade to the latest CDH 5 release and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

To upgrade Oozie to the latest CDH 5 release, proceed as follows.

Step 1: Back Up the Configuration

Back up the Oozie configuration files in `/etc/oozie` and the Oozie database.

For convenience you may want to save Oozie configuration files in your home directory; you will need them after installing the new version of Oozie.

Step 2: Stop the Oozie Server.

To stop the Oozie Server:

```
sudo service oozie stop
```

Step 3: Install Oozie

Follow the procedure under Installing Oozie on page 300 and then proceed to Configuring Oozie after Upgrading from an Earlier CDH 5 Release on page 305.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Installing Oozie

Oozie is distributed as two separate packages; a client package (`oozie-client`) and a server package (`oozie`). Depending on what you are planning to install, choose the appropriate packages and install them using your preferred package manager application.

Note:

The Oozie server package, `oozie`, is preconfigured to work with MRv2 (YARN). To configure the Oozie server to work with MRv1, see Configuring the Hadoop Version to Use.

Important:

If you have not already done so, install Cloudera's Yum, `zypper/YaST` or Apt repository before using the following commands to install Oozie. For instructions, see CDH 5 Installation.

To install the Oozie server package on an Ubuntu and other Debian system:

```
$ sudo apt-get install oozie
```

To install the Oozie client package on an Ubuntu and other Debian system:

```
$ sudo apt-get install oozie-client
```

To install the Oozie server package on a Red Hat-compatible system:

```
$ sudo yum install oozie
```

To install the Oozie client package on a Red Hat-compatible system:

```
$ sudo yum install oozie-client
```

To install the Oozie server package on a SLES system:

```
$ sudo zypper install oozie
```

To install the Oozie client package on a SLES system:

```
$ sudo zypper install oozie-client
```

Note:

Installing the `oozie` package creates an `oozie` service configured to start Oozie at system startup time.

You are now ready to configure Oozie. See the next section.

Configuring Oozie

This section explains how to configure which Hadoop version to use, and provides separate procedures for each of the following:

- Configuring Oozie after Upgrading from CDH 4 on page 302
- Configuring Oozie after Upgrading from an Earlier CDH 5 Release on page 305
- Configuring Oozie after a New Installation on page 308.

Configuring which Hadoop Version to Use

The Oozie client does not interact directly with Hadoop MapReduce, and so it does not require any MapReduce configuration.

The Oozie server can work with either MRv1 or YARN. **It cannot work with both simultaneously.**

You set the MapReduce version the Oozie server works with by means of the `alternatives` command (or `update-alternatives`, depending on your operating system). As well as distinguishing between YARN and MRv1, the commands differ depending on whether or not you are using SSL.

- To use YARN (without SSL):

  ```
  alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.http
  ```

- To use YARN (with SSL):

  ```
  alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.https
  ```

- To use MRv1(without SSL) :

```
alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.http.mr1
```

- To use MRv1(with SSL) :

```
alternatives --set oozie-tomcat-conf /etc/oozie/tomcat-conf.https.mr1
```

> **Important: If you are upgrading from a release earlier than CDH 5 Beta 2**
>
> In earlier releases, the mechanism for setting the MapReduce version was the `CATALINA_BASE` variable in `/etc/oozie/conf/oozie-env.sh`. This does not work as of CDH 5 Beta 2, and in fact could cause problems. Check your `/etc/oozie/conf/oozie-env.sh` and make sure you have the new version. The new version contains the line:
>
> ```
> export CATALINA_BASE=/var/lib/oozie/tomcat-deployment
> ```

Configuring Oozie after Upgrading from CDH 4

> **Note:**
>
> If you are installing Oozie for the first time, skip this section and proceed with Configuring Oozie after a New Installation.

Step 1: Update Configuration Files

1. Edit the new Oozie CDH 5 `oozie-site.xml`, and set all customizable properties to the values you set in the CDH 4 `oozie-site.xml`:

   > **Important:**
   >
   > DO NOT copy over the CDH 4 configuration files into the CDH 5 configuration directory.

2. If necessary do the same for the `oozie-log4j.properties`, `oozie-env.sh` and the `adminusers.txt` files.

Step 2: Upgrade the Database

> **Important:**
>
> - Do not proceed before you have edited the configuration files as instructed in Step 1.
> - Before running the database upgrade tool, copy or symbolically link the JDBC driver JAR for the database you are using into the `/var/lib/oozie/` directory.

Oozie CDH 5 provides a command-line tool to perform the database schema and data upgrade that is required when you upgrade Oozie from CDH 4 to CDH 5. The tool uses Oozie configuration files to connect to the database and perform the upgrade.

The database upgrade tool works in two modes: it can do the upgrade in the database or it can produce an SQL script that a database administrator can run manually. If you use the tool to perform the upgrade, you must do it as a database user who has permissions to run DDL operations in the Oozie database.

- **To run the Oozie database upgrade tool against the database:**

Important:

This step must be done as the oozie Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh upgrade -run
```

You will see output such as this (the output of the script may differ slightly depending on the database vendor):

```
Validate DB Connection
DONE
Check DB schema exists
DONE
Verify there are not active Workflow Jobs
DONE
Check OOZIE_SYS table does not exist
DONE
Get Oozie DB version
DONE
Upgrade SQL schema
DONE
Upgrading to db schema for Oozie 4.0
Update db.version in OOZIE_SYS table to 2
DONE
Post-upgrade COORD_JOBS new columns default values
DONE
Post-upgrade COORD_JOBS & COORD_ACTIONS status values
DONE
Post-upgrade MISSING_DEPENDENCIES column in Derby
DONE
Table 'WF_ACTIONS' column 'execution_path', length changed to 1024
Table 'WF_ACTIONS, column 'error_message', changed to varchar/varchar2
Table 'COORD_JOB' column 'frequency' changed to varchar/varchar2
DONE
Post-upgrade BUNDLE_JOBS, COORD_JOBS, WF_JOBS to drop AUTH_TOKEN column
DONE
Upgrading to db schema for Oozie 4.0.0-cdh5.0.0
Update db.version in OOZIE_SYS table to 3
DONE
Dropping discriminator column
DONE

Oozie DB has been upgraded to Oozie version '4.0.0-cdh5.0.0’

The SQL commands have been written to: /tmp/ooziedb-3809837539907706.sql
```

- **To create the upgrade script:**

Important:

This step must be done as the oozie Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh upgrade -sqlfile <SCRIPT>
```

For example:

```
$ sudo -u oozie /usr/lib/bin/ooziedb.sh upgrade -sqlfile oozie-upgrade.sql
```

You should see output such as the following (the output of the script may differ slightly depending on the database vendor):

```
Validate DB Connection
DONE
Check DB schema exists
DONE
Verify there are not active Workflow Jobs
DONE
Check OOZIE_SYS table does not exist
DONE
Get Oozie DB version
DONE
Upgrade SQL schema
DONE
Upgrading to db schema for Oozie 4.0
Update db.version in OOZIE_SYS table to 2
DONE
Post-upgrade COORD_JOBS new columns default values
DONE
Post-upgrade COORD_JOBS & COORD_ACTIONS status values
DONE
Post-upgrade MISSING_DEPENDENCIES column in Derby
DONE
Table 'WF_ACTIONS' column 'execution_path', length changed to 1024
Table 'WF_ACTIONS, column 'error_message', changed to varchar/varchar2
Table 'COORD_JOB' column 'frequency' changed to varchar/varchar2
DONE
Post-upgrade BUNDLE_JOBS, COORD_JOBS, WF_JOBS to drop AUTH_TOKEN column
DONE
Upgrading to db schema for Oozie 4.0.0-cdh5.0.0
Update db.version in OOZIE_SYS table to 3
DONE
Dropping discriminator column
DONE

The SQL commands have been written to: oozie-upgrade.sql

WARN: The SQL commands have NOT been executed, you must use the '-run' option
```

- **Important:**

  If you used the `-sqlfile` option instead of `-run`, Oozie database schema has not been upgraded. You need to run the `oozie-upgrade` script against your database.

Step 3: Upgrade the Oozie Shared Library

- **Important:**

  This step is required; CDH 5 Oozie does not work with CDH 4 shared libraries.

CDH 5 Oozie has a new shared library which bundles CDH 5 JAR files for streaming, DistCp and for Pig, Hive, HiveServer 2, Sqoop, and HCatalog.

- **Note:**

  The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:

  - The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.
  - The shared library file for MRv1 is `oozie-sharelib-mr1.tar.gz`.

Proceed as follows to upgrade the shared library.

1. Delete the Oozie shared libraries from HDFS. For example:

   ```
   $ sudo -u oozie hadoop fs -rmr /user/oozie/share
   ```

   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>`
   - If the current shared libraries are in another location, make sure you use this other location when you run the above command(s).

2. install the Oozie CDH 5 shared libraries. For example:

   ```
   $ sudo oozie-setup sharelib create -fs <FS_URI> -locallib
   /usr/lib/oozie/oozie-sharelib-yarn.tar.gz
   ```

   where *FS_URI* is the HDFS URI of the filesystem that the shared library should be installed on (for example, `hdfs://<HOST>:<PORT>`).

   Important:

   If you are installing Oozie to work with MRv1, make sure you use `oozie-sharelib-mr1.tar.gz` instead.

Step 4: Start the Oozie Server

Now you can start Oozie:

```
$ sudo service oozie start
```

Check Oozie's `oozie.log` to verify that Oozie has started successfully.

Step 5: Upgrade the Oozie Client

Although older Oozie clients work with the new Oozie server, you need to install the new version of the Oozie client in order to use all the functionality of the Oozie server.

To upgrade the Oozie client, if you have not already done so, follow the steps under Installing Oozie.

Configuring Oozie after Upgrading from an Earlier CDH 5 Release

Note:

If you are installing Oozie for the first time, skip this section and proceed with Configuring Oozie after a New Installation on page 308.

Step 1: Update Configuration Files

1. Edit the new Oozie CDH 5 `oozie-site.xml`, and set all customizable properties to the values you set in the previous `oozie-site.xml`.
2. If necessary do the same for the `oozie-log4j.properties`, `oozie-env.sh` and the `adminusers.txt` files.

Step 2: Upgrade the Database

Important:

- Do not proceed before you have edited the configuration files as instructed in Step 1.
- Before running the database upgrade tool, copy or symbolically link the JDBC driver JAR for the database you are using into the `/var/lib/oozie/` directory.

Oozie CDH 5 provides a command-line tool to perform the database schema and data upgrade. The tool uses Oozie configuration files to connect to the database and perform the upgrade.

The database upgrade tool works in two modes: it can do the upgrade in the database or it can produce an SQL script that a database administrator can run manually. If you use the tool to perform the upgrade, you must do it as a database user who has permissions to run DDL operations in the Oozie database.

- **To run the Oozie database upgrade tool against the database:**

  Important:

  This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

  ```
  $ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh upgrade -run
  ```

  You will see output such as this (the output of the script may differ slightly depending on the database vendor):

  ```
  Validate DB Connection
  DONE
  Check DB schema exists
  DONE
  Verify there are not active Workflow Jobs
  DONE
  Check OOZIE_SYS table does not exist
  DONE
  Get Oozie DB version
  DONE
  Upgrade SQL schema
  DONE
  Upgrading to db schema for Oozie 4.0.0-cdh5.0.0
  Update db.version in OOZIE_SYS table to 3
  DONE
  Converting text columns to bytea for all tables
  DONE
  Get Oozie DB version
  DONE

  Oozie DB has been upgraded to Oozie version '4.0.0-cdh5.0.0'

  The SQL commands have been written to: /tmp/ooziedb-8676029205446760413.sql
  ```

- **To create the upgrade script:**

  Important:

  This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

  ```
  $ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh upgrade -sqlfile <SCRIPT>
  ```

For example:

```
$ sudo -u oozie /usr/lib/bin/ooziedb.sh upgrade -sqlfile oozie-upgrade.sql
```

You should see output such as the following (the output of the script may differ slightly depending on the database vendor):

```
Validate DB Connection
DONE
Check DB schema exists
DONE
Verify there are not active Workflow Jobs
DONE
Check OOZIE_SYS table does not exist
DONE
Get Oozie DB version
DONE
Upgrade SQL schema
DONE
Upgrading to db schema for Oozie 4.0.0-cdh5.0.0
Update db.version in OOZIE_SYS table to 3
DONE
Converting text columns to bytea for all tables
DONE
Get Oozie DB version
DONE

The SQL commands have been written to: oozie-upgrade.sql

WARN: The SQL commands have NOT been executed, you must use the '-run' option
```

Important:

If you used the `-sqlfile` option instead of `-run`, Oozie database schema has not been upgraded. You need to run the `oozie-upgrade` script against your database.

Step 3: Upgrade the Oozie Shared Library

Important:

This step is required; the current version of Oozie does not work with shared libraries from an earlier version.

The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:

- The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.
- The shared library file for MRv1 is `oozie-sharelib-mr1.tar.gz`.

To upgrade the shared library, proceed as follows.

1. Delete the Oozie shared libraries from HDFS. For example:

```
$ sudo -u oozie hadoop fs -rmr /user/oozie/share
```

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>`
- If the current shared libraries are in another location, make sure you use this other location when you run the above command(s).

2. install the Oozie CDH 5 shared libraries. For example:

```
$ sudo oozie-setup sharelib create -fs <FS_URI> -locallib
/usr/lib/oozie/oozie-sharelib-yarn.tar.gz
```

where *FS_URI* is the HDFS URI of the filesystem that the shared library should be installed on (for example, `hdfs://<HOST>:<PORT>`).

Important:

If you are installing Oozie to work with MRv1, make sure you use `oozie-sharelib-mr1.tar.gz` instead.

Step 4: Start the Oozie Server

Now you can start Oozie:

```
$ sudo service oozie start
```

Check Oozie's `oozie.log` to verify that Oozie has started successfully.

Step 5: Upgrade the Oozie Client

Although older Oozie clients work with the new Oozie server, you need to install the new version of the Oozie client in order to use all the functionality of the Oozie server.

To upgrade the Oozie client, if you have not already done so, follow the steps under Installing Oozie on page 300.

Configuring Oozie after a New Installation

Note:

Follow the instructions in this section if you are installing Oozie for the first time. If you are upgrading Oozie from CDH 4 or from an earlier CDH 5 release, skip this subsection and choose the appropriate instructions earlier in this section: Configuring Oozie after Upgrading from CDH 4 on page 302 or Configuring Oozie after Upgrading from an Earlier CDH 5 Release on page 305.

When you install Oozie from an RPM or Debian package, Oozie server creates all configuration, documentation, and runtime files in the standard Linux directories, as follows.

Type of File	Where Installed
binaries	`/usr/lib/oozie/`
configuration	`/etc/oozie/conf/`
documentation	for SLES: `/usr/share/doc/packages/oozie/` for other platforms: `/usr/share/doc/oozie/`
examples TAR.GZ	for SLES: `/usr/share/doc/packages/oozie/` for other platforms: `/usr/share/doc/oozie/`
sharelib TAR.GZ	`/usr/lib/oozie/`
data	`/var/lib/oozie/`
logs	`/var/log/oozie/`
temp	`/var/tmp/oozie/`
PID file	`/var/run/oozie/`

Deciding which Database to Use

Oozie has a built-in Derby database, but Cloudera recommends that you use a Postgres, MySQL, or Oracle database instead, for the following reasons:

- Derby runs in embedded mode and it is not possible to monitor its health.
- It is not clear how to implement a live backup strategy for the embedded Derby database, though it may be possible.
- Under load, Cloudera has observed locks and rollbacks with the embedded Derby database which don't happen with server-based databases.

Configuring Oozie to Use PostgreSQL

Use the procedure that follows to configure Oozie to use PostgreSQL instead of Apache Derby.

Step 1: Install PostgreSQL 8.4.x or 9.0.x.

Note:

See CDH 5 Requirements and Supported Versions for tested versions.

Step 2: Create the Oozie user and Oozie database.

For example, using the PostgreSQL `psql` command-line tool:

```
$ psql -U postgres
Password for user postgres: *****

postgres=# CREATE ROLE oozie LOGIN ENCRYPTED PASSWORD 'oozie'
 NOSUPERUSER INHERIT CREATEDB NOCREATEROLE;
CREATE ROLE
```

```
postgres=# CREATE DATABASE "oozie" WITH OWNER = oozie
 ENCODING = 'UTF8'
 TABLESPACE = pg_default
 LC_COLLATE = 'en_US.UTF8'
 LC_CTYPE = 'en_US.UTF8'
 CONNECTION LIMIT = -1;
CREATE DATABASE

postgres=# \q
```

Step 3: Configure PostgreSQL to accept network connections for user oozie.

1. Edit the `postgresql.conf` file and set the `listen_addresses` property to `*`, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the `standard_conforming_strings` property is set to `off`.
2. Edit the PostgreSQL `data/pg_hba.conf` file as follows:

```
host    oozie         oozie         0.0.0.0/0             md5
```

Edit the PostgreSQL `data/pg_hba.conf` file as follows:

```
host    oozie         oozie         0.0.0.0/0             md5
```

Step 4: Reload the PostgreSQL configuration.

```
$ sudo -u postgres pg_ctl reload -s -D /opt/PostgreSQL/8.4/data
```

Step 5: Configure Oozie to use PostgreSQL

Edit the `oozie-site.xml` file as follows:

```
...
    <property>
        <name>oozie.service.JPAService.jdbc.driver</name>
        <value>org.postgresql.Driver</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.url</name>
        <value>jdbc:postgresql://localhost:5432/oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

Note:

In the JDBC URL property, replace `localhost` with the hostname where PostgreSQL is running.

In the case of PostgreSQL, unlike MySQL or Oracle, there is no need to download and install the JDBC driver separately, as it is license-compatible with Oozie and bundled with it.

Configuring Oozie to Use MySQL

Use the procedure that follows to configure Oozie to use MySQL instead of Apache Derby.

Step 1: Install and start MySQL 5.x

Note:

See CDH 5 Requirements and Supported Versions for tested versions.

Step 2: Create the Oozie database and Oozie MySQL user.

For example, using the MySQL `mysql` command-line tool:

```
$ mysql -u root -p
Enter password: ******

mysql> create database oozie;
Query OK, 1 row affected (0.03 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'localhost' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql>  grant all privileges on oozie.* to 'oozie'@'%' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> exit
Bye
```

Step 3: Configure Oozie to use MySQL.

Edit properties in the `oozie-site.xml` file as follows:

```
...
    <property>
        <name>oozie.service.JPAService.jdbc.driver</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.url</name>
        <value>jdbc:mysql://localhost:3306/oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

Note:

In the JDBC URL property, replace `localhost` with the hostname where MySQL is running.

Step 4: Add the MySQL JDBC driver JAR to Oozie.

Copy or symbolically link the MySQL JDBC driver JAR into the `/var/lib/oozie/` directory.

Note:

You must manually download the MySQL JDBC driver JAR file.

Configuring Oozie to use Oracle

Use the procedure that follows to configure Oozie to use Oracle 11g instead of Apache Derby.

> **Note:**
>
> See CDH 5 Requirements and Supported Versions for tested versions.

Step 1: Install and start Oracle 11g.

Use Oracle's instructions.

Step 2: Create the Oozie Oracle user.

For example, using the Oracle `sqlplus` command-line tool:

```
$ sqlplus system@localhost

Enter password: ******

SQL> create user oozie identified by oozie default tablespace users temporary tablespace
 temp;

User created.

SQL> grant all privileges to oozie;

Grant succeeded.

SQL> exit

$
```

Step 3: Configure Oozie to use Oracle.

Edit the `oozie-site.xml` file as follows.

```
...
    <property>
        <name>oozie.service.JPAService.jdbc.driver</name>
        <value>oracle.jdbc.OracleDriver</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.url</name>
        <value>jdbc:oracle:thin:@//myhost:1521/oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>
        <name>oozie.service.JPAService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

> **Note:**
>
> In the JDBC URL property, replace `myhost` with the hostname where Oracle is running and replace `oozie` with the TNS name of the Oracle database.

Step 4: Add the Oracle JDBC driver JAR to Oozie.

Copy or symbolically link the Oracle JDBC driver JAR into the `/var/lib/oozie/` directory.

Note:

You must manually download the Oracle JDBC driver JAR file.

Creating the Oozie Database Schema

After configuring Oozie database information and creating the corresponding database, create the Oozie database schema. Oozie provides a database tool for this purpose.

Note:

The Oozie database tool uses Oozie configuration files to connect to the database to perform the schema creation; before you use the tool, make you have created a database and configured Oozie to work with it as described above.

The Oozie database tool works in 2 modes: it can create the database, or it can produce an SQL script that a database administrator can run to create the database manually. If you use the tool to create the database schema, you must have the permissions needed to execute DDL operations.

To run the Oozie database tool against the database

Important:

This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh create -run
```

You should see output such as the following (the output of the script may differ slightly depending on the database vendor) :

```
Validate DB Connection.
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE

Oozie DB has been created for Oozie version '4.0.0-cdh5.0.0'

The SQL commands have been written to: /tmp/ooziedb-5737263881793872034.sql
```

To create the upgrade script

Important:

This step must be done as the oozie Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

Run `/usr/lib/oozie/bin/ooziedb.sh create -sqlfile <SCRIPT>`. For example:

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh create -sqlfile oozie-create.sql
```

You should see output such as the following (the output of the script may differ slightly depending on the database vendor) :

```
Validate DB Connection.
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE

Oozie DB has been created for Oozie version '4.0.0-cdh5.0.0'

The SQL commands have been written to: oozie-create.sql

WARN: The SQL commands have NOT been executed, you must use the '-run' option
```

Important:

If you used the `-sqlfile` option instead of `-run`, Oozie database schema has not been created. You need to run the `oozie-create.sql` script against your database.

Enabling the Oozie Web Console

To enable Oozie's web console, you must download and add the ExtJS library to the Oozie server. *If you have not already done this,* proceed as follows.

Step 1: Download the Library

Download the ExtJS version 2.2 library from http://archive.cloudera.com/gplextras/misc/ext-2.2.zip and place it a convenient location.

Step 2: Install the Library

Extract the `ext-2.2.zip` file into `/var/lib/oozie`.

Configuring Oozie with Kerberos Security

To configure Oozie with Kerberos security, see Oozie Security Configuration.

Installing the Oozie Shared Library in Hadoop HDFS

The Oozie installation bundles the Oozie shared library, which contains all of the necessary JARs to enable workflow jobs to run streaming, DistCp, Pig, Hive, and Sqoop actions.

The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:

- The shared library file for MRv1 is `oozie-sharelib-mr1.tar.gz`.
- The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.

Important:

If Hadoop is configured with Kerberos security enabled, you must first configure Oozie with Kerberos Authentication. For instructions, see Oozie Security Configuration. Before running the commands in the following instructions, you must run the `sudo -u oozie kinit -k -t /etc/oozie/oozie.keytab` and `kinit -k hdfs` commands. Then, instead of using commands in the form `sudo -u <user> <command>`, use just `<command>`; for example, `$ hadoop fs -mkdir /user/oozie`

To install the Oozie shared library in Hadoop HDFS in the oozie user home directory

```
$ sudo -u hdfs hadoop fs -mkdir /user/oozie
$ sudo -u hdfs hadoop fs -chown oozie:oozie /user/oozie
$ sudo oozie-setup sharelib create -fs <FS_URI> -locallib
/usr/lib/oozie/oozie-sharelib-yarn.tar.gz
```

where *FS_URI* is the HDFS URI of the filesystem that the shared library should be installed on (for example, `hdfs://<HOST>:<PORT>`).

Important:

If you are installing Oozie to work with MRv1 use `oozie-sharelib-mr1.tar.gz` instead.

Configuring Support for Oozie Uber JARs

An ***uber JAR*** is a JAR that contains other JARs with dependencies in a `lib/` folder inside the JAR. You can configure the cluster to handle uber JARs properly for the MapReduce action (as long as it does not include any streaming or pipes) by setting the following property in the `oozie-site.xml` file:

```
...
    <property>
        <name>oozie.action.mapreduce.uber.jar.enable</name>
    <value>true</value>

    ...
```

When this property is set, users can use the `oozie.mapreduce.uber.jar` configuration property in their MapReduce workflows to notify Oozie that the specified JAR file is an uber JAR.

Configuring Oozie to Run against a Federated Cluster

To run Oozie against a federated HDFS cluster using ViewFS, configure the `oozie.service.HadoopAccessorService.supported.filesystems property` in `oozie-site.xml` as follows:

```
<property>
     <name>oozie.service.HadoopAccessorService.supported.filesystems</name>
     <value>hdfs,viewfs</value>
</property>
```

Starting, Stopping, and Accessing the Oozie Server

Starting the Oozie Server

After you have completed *all* of the required configuration steps, you can start Oozie:

```
$ sudo service oozie start
```

If you see the message `Oozie System ID [oozie-oozie] started` in the `oozie.log` log file, the system has started successfully.

Note:

By default, Oozie server runs on port `11000` and its URL is `http://<OOZIE_HOSTNAME>:11000/oozie`.

Stopping the Oozie Server

```
$ sudo service oozie stop
```

Accessing the Oozie Server with the Oozie Client

The Oozie client is a command-line utility that interacts with the Oozie server via the Oozie web-services API.

Use the `/usr/bin/oozie` script to run the Oozie client.

For example, if you want to invoke the client on the same machine where the Oozie server is running:

```
$ oozie admin -oozie http://localhost:11000/oozie -status
System mode: NORMAL
```

To make it convenient to use this utility, set the environment variable `OOZIE_URL` to point to the URL of the Oozie server. Then you can skip the `-oozie` option.

For example, if you want to invoke the client on the same machine where the Oozie server is running, set the `OOZIE_URL` to `http://localhost:11000/oozie`.

```
$ export OOZIE_URL=http://localhost:11000/oozie
$ oozie admin -version
Oozie server build version: 4.0.0-cdh5.0.0
```

Important:

If Oozie is configured with Kerberos Security enabled:

- You must have a Kerberos session running. For example, you can start a session by running the `kinit` command.
- **Do not** use `localhost` as in the above examples.

As with every service that uses Kerberos, Oozie has a Kerberos *principal* in the form `<SERVICE>/<HOSTNAME>@<REALM>`. In a Kerberos configuration, you **must** use the `<HOSTNAME>` value in the Kerberos principal to specify the Oozie server; for example, if the `<HOSTNAME>` in the principal is `myoozieserver.mydomain.com`, set `OOZIE_URL` as follows:

```
$ export OOZIE_URL=http://myoozieserver.mydomain.com:11000/oozie
```

If you use an alternate hostname or the IP address of the service, Oozie will not work properly.

Accessing the Oozie Server with a Browser

If you have enabled the Oozie web console by adding the ExtJS library, you can connect to the console at `http://<OOZIE_HOSTNAME>:11000/oozie`.

> **Note:**
>
> If the Oozie server is configured to use Kerberos HTTP SPNEGO Authentication, you must use a web browser that supports Kerberos HTTP SPNEGO (for example, Firefox or Internet Explorer).

Configuring Oozie Failover (hot/cold)

> **Note:**
>
> The functionality described below is supported in CDH 5, but Cloudera recommends that you use the new capabilities introduced in CDH 5 instead.

1. Set up your database for High Availability (see the database documentation for details).

   > **Note:**
   >
   > Oozie database configuration properties may need special configuration (see the JDBC driver documentation for details).

2. Configure Oozie on two or more servers:
3. These servers should be configured identically
4. Set the `OOZIE_HTTP_HOSTNAME` variable in `oozie-env.sh` to the Load Balancer or Virtual IP address (see step 3)
5. Only one of the Oozie servers should be started (the *hot* server).
6. Use either a Virtual IP Address or Load Balancer to direct traffic to the hot server.
7. Access Oozie via the Virtual IP or Load Balancer address.

Points to note

- The Virtual IP Address or Load Balancer can be used to periodically check the health of the hot server.
- If something is wrong, you can shut down the hot server, start the cold server, and redirect the Virtual IP Address or Load Balancer to the new hot server.
- This can all be automated with a script, but a false positive indicating the hot server is down will cause problems, so test your script carefully.
- There will be no data loss.
- Any running workflows will continue from where they left off.
- It takes only about 15 seconds to start the Oozie server.

See also Configuring Oozie to Use HDFS HA.

Viewing the Oozie Documentation

For additional Oozie documentation, see http://archive.cloudera.com/cdh5/cdh/5/oozie/.

Pig Installation

Apache Pig enables you to analyze large amounts of data using Pig's query language called Pig Latin. Pig Latin queries run in a distributed way on a Hadoop cluster.

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install or upgrade Pig. For instructions, see CDH 5 Installation on page 27.

Use the following sections to install or upgrade Pig:

- Upgrading Pig
- Installing Pig
- Using Pig with HBase
- Installing DataFu
- Apache Pig Documentation

Upgrading Pig

Note:

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

Upgrading Pig from CDH 4 to CDH 5

To upgrade Pig to CDH 5:

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5, you can skip Step 1 below and proceed with installing the new CDH 5 version of Pig.

Step 1: Remove Pig

1. Exit the Grunt shell and make sure no Pig scripts are running.
2. Remove the CDH 4 version of Pig

To remove Pig On Red Hat-compatible systems:

```
$ sudo yum remove hadoop-pig
```

To remove Pig on SLES systems:

```
$ sudo zypper remove pig
```

To remove Pig on Ubuntu and other Debian systems:

```
$ sudo apt-get remove pig
```

Step 2: Install the new version

Follow the instructions in the next section, Installing Pig.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Upgrading Pig from an Earlier CDH 5 release

The instructions that follow assume that you are upgrading Pig as part of a CDH 5 upgrade, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

To upgrade Pig from an earlier CDH 5 release:

1. Exit the Grunt shell and make sure no Pig scripts are running.
2. Install the new version, following the instructions in the next section, Installing Pig on page 320.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Installing Pig

To install Pig On Red Hat-compatible systems:

```
$ sudo yum install pig
```

To install Pig on SLES systems:

```
$ sudo zypper install pig
```

To install Pig on Ubuntu and other Debian systems:

```
$ sudo apt-get install pig
```

Note:

Pig automatically uses the active Hadoop configuration (whether standalone, pseudo-distributed mode, or distributed). After installing the Pig package, you can start Pig.

To start Pig in interactive mode (YARN)

Important:

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

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

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

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

To start Pig, use the following command.

```
$ pig
```

To start Pig in interactive mode (MRv1)

Use the following command:

```

$ pig
```

You should see output similar to the following:

```
2012-02-08 23:39:41,819 [main] INFO  org.apache.pig.Main - Logging error messages to:
 /home/arvind/pig-0.11.0-cdh5b1/bin/pig_1328773181817.log
2012-02-08 23:39:41,994 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop
 file system at: hdfs://localhost/
...
grunt>
```

Examples

To verify that the input and output directories from the YARN or MRv1 example `grep` job exist, list an HDFS directory from the Grunt Shell:

```
grunt> ls
hdfs://localhost/user/joe/input <dir>
hdfs://localhost/user/joe/output <dir>
```

To run a grep example job using Pig for grep inputs:

```
grunt> A = LOAD 'input';
grunt> B = FILTER A BY $0 MATCHES '.*dfs[a-z.]+.*';
grunt> DUMP B;
```

Note:

To check the status of your job while it is running, look at the ResourceManager web console (YARN) or JobTracker web console (MRv1).

Using Pig with HBase

To allow Pig scripts to use HBase, add the following statement to the top of each script. Replace the <component_version> strings with the current HBase, ZooKeeper and CDH version numbers.

```
register /usr/lib/zookeeper/zookeeper-<ZooKeeper_version>-cdh<CDH_version>.jar
register /usr/lib/hbase/hbase-<HBase_version>-cdh<CDH_version>-security.jar
```

For example,

```
register /usr/lib/zookeeper/zookeeper-3.4.5-cdh5.0.0.jar
register /usr/lib/hbase/hbase-0.95.2-cdh5.0.0-security.jar
```

Installing DataFu

DataFu is a collection of Apache Pig UDFs (User-Defined Functions) for statistical evaluation that were developed by LinkedIn and have now been open sourced under an Apache 2.0 license.

To use DataFu:

1. Install the DataFu package:

Operating system	Install command
Red-Hat-compatible	`sudo yum install pig-udf-datafu`
SLES	`sudo zypper install pig-udf-datafu`
Debian or Ubuntu	`sudo apt-get install pig-udf-datafu`

   This puts the `datafu-0.0.4-cdh5.0.0.jar` file in `/usr/lib/pig`.

2. Register the JAR. Replace the `<component_version>` string with the current DataFu and CDH version numbers.

```
REGISTER /usr/lib/pig/datafu-<DataFu_version>-cdh<CDH_version>.jar
```

   For example,

```
REGISTER /usr/lib/pig/datafu-0.0.4-cdh5.0.0.jar
```

A number of usage examples and other information are available at https://github.com/linkedin/datafu.

Viewing the Pig Documentation

For additional Pig documentation, see http://archive.cloudera.com/cdh5/cdh/5/pig.

Search Installation

This documentation describes how to install Cloudera Search powered by Solr. This documentation also explain how to install and start supporting tools and services such as the ZooKeeper Server, MapReduce tools for use with Cloudera Search, and Flume Solr Sink.

After completing the process of installing Cloudera Search, as described in this document, proceed to configuring and using Cloudera Search as described in the Cloudera Search User Guide. The user guide includes the Cloudera Search Tutorial, as well as topics describing extracting, transforming, and loading data, establishing high availability, and troubleshooting.

Cloudera Search documentation includes:

- CDH 5 Release Notes
- CDH Version and Packaging Information
- Cloudera Search User Guide
- Cloudera Search Frequently Asked Questions

Sentry Installation

Sentry enables role-based, fine-grained authorization for HiveServer2 and Cloudera Impala. It provides classic database-style authorization for Hive and Impala. For more information, and instructions on configuring Sentry for Hive, see Sentry Policy File Configuration.

Installing and Upgrading Sentry

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands. For instructions, see CDH 5 Installation.

Upgrading Sentry from CDH 4 to CDH 5

To upgrade Sentry from CDH 4 to CDH 5, you must uninstall the old version and install the new version. If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5 on page 57, you can skip Step 1 below and proceed with installing the new CDH 5 version of Sentry.

1. **Remove the CDH 4 Version of Sentry**

   Remove Sentry as follows, depending on your operating system:

OS	Command
RHEL	`$ sudo yum remove sentry`
SLES	`$ sudo zypper remove sentry`
Ubuntu or Debian	`$ sudo apt-get remove sentry`

2. **Install the New Version of Sentry**

   Follow instructions in the next section to install the CDH 5 version of Sentry.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

The upgrade is now complete.

Installing Sentry

Install Sentry as follows, depending on your operating system:

OS	Command
RHEL	`$ sudo yum install sentry`
SLES	`$ sudo zypper install sentry`

Ubuntu or Debian	`$ sudo apt-get update;`
	`$ sudo apt-get install sentry`

Snappy Installation

Snappy is a compression/decompression library. It aims for very high speeds and reasonable compression, rather than maximum compression or compatibility with other compression libraries. Use the following sections to install, upgrade, and use Snappy.

- Upgrading Snappy
- Installing Snappy
- Using Snappy for MapReduce Compression
- Using Snappy for Pig Compression
- Using Snappy for Hive Compression
- Using Snappy Compression in Sqoop Imports
- Using Snappy Compression with HBase
- Apache Snappy Documentation

Upgrading Snappy

To upgrade Snappy, simply install the hadoop package if you haven't already done so. This applies whether you are upgrading from CDH 4 or from an earlier CDH 5 release.

Note:

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

Installing Snappy

Snappy is provided in the hadoop package along with the other native libraries (such as native gzip compression).

To take advantage of Snappy compression you need to set certain configuration properties, which are explained in the following sections.

Using Snappy for MapReduce Compression

It's very common to enable MapReduce intermediate compression, since this can make jobs run faster without you having to make any application changes. Only the temporary intermediate files created by Hadoop for the shuffle phase are compressed (the final output may or may not be compressed). Snappy is ideal in this case because it compresses and decompresses very fast compared to other compression algorithms, such as Gzip.

To enable Snappy for MapReduce intermediate compression for the whole cluster, set the following properties in `mapred-site.xml`:

- For MRv1:

```
<property>
  <name>mapred.compress.map.output</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compression.codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

- For YARN:

```
<property>
  <name>mapreduce.map.output.compress</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compress.codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

You can also set these properties on a per-job basis.

Use the properties in the following table to compress the final output of a MapReduce job. These are usually set on a per-job basis.

MRv1 Property	YARN Property	Description
`mapred.output.compress`	`mapreduce.output.fileoutputformat.compress`	Whether to compress the final job outputs (`true` or `false`)
`mapred.output.compression.codec`	`mapreduce.output.fileoutputformat.compress.codec`	If the final job outputs are to be compressed, which codec should be used. Set to `org.apache.hadoop.io.compress.SnappyCodec` for Snappy compression.
`mapred.output.compression.type`	`mapreduce.output.fileoutputformat.compress.type`	For `SequenceFile` outputs, what type of compression should be used (`NONE`, `RECORD`, or `BLOCK`). `BLOCK` is recommended.

- **Note:**

  The MRv1 property names are also supported (though deprecated) in MRv2 (YARN), so it's not mandatory to update them in this release.

Using Snappy for Pig Compression

Set the same properties for Pig as for MapReduce (see the table in the previous section).

Using Snappy for Hive Compression

To enable Snappy compression for Hive output when creating `SequenceFile` outputs, use the following settings:

```
SET hive.exec.compress.output=true;
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;
SET mapred.output.compression.type=BLOCK;
```

Using Snappy compression in Sqoop 1 and Sqoop 2 Imports

- **For Sqoop 1:**

On the command line, use the following option to enable Snappy compression:

```
--compression-codec org.apache.hadoop.io.compress.SnappyCodec
```

It is a good idea to use the `--as-sequencefile` option with this compression option.

- **For Sqoop 2:**

  When you create a job (`sqoop:000> create job`), choose `7` (`SNAPPY`) as the compression format.

Using Snappy Compression with HBase

If you install Hadoop and HBase from RPM or Debian packages, Snappy requires no HBase configuration.

Viewing the Snappy Documentation

For more information about Snappy, see http://code.google.com/p/snappy/.

Spark Installation

Spark is a fast, general engine for large-scale data processing.

The following sections describe how to install and configure Spark.

- Spark Packaging on page 333
- Spark Prerequisites on page 333
- Installing and Upgrading Spark on page 333
- Configuring and Running Spark (Standalone Mode) on page 334

See also the Apache Spark Documentation.

Spark Packaging

The packaging options for installing Spark are:

- RPM packages
- Debian packages

There are five Spark packages:

- `spark-core`: delivers core functionality of spark
- `spark-worker`: `init` scripts for `spark-worker`
- `spark-master`: `init` scripts for `spark-master`
- `spark-python`: Python client for Spark
- `spark-history-server`

Spark Prerequisites

- An operating system supported by CDH 5
- Oracle JDK
- The `hadoop-client` package (see Installing CDH 5 on page 27)

Installing and Upgrading Spark

To install or upgrade the Spark packages on a Red-Hat-compatible system:

Note: Install Cloudera Repository

Before installing Spark, install the Cloudera `yum`, `zypper/YaST` or `apt` repository.

To see which version of Spark is shipping in the current release, check the CDH Version and Packaging Information. For important information, see the CDH 5 Release Notes, in particular:

- New Features in CDH 5
- Apache Spark Incompatible Changes
- Apache Spark Known Issues

To install or upgrade the Spark packages on an Red-Hat-compatible system:

```
$ sudo yum install spark-core spark-master spark-worker spark-history-server spark-python
```

To install or upgrade the Spark packages on a SLES system:

```
$ sudo zypper install spark-core spark-master spark-worker spark-history-server
spark-python
```

To install or upgrade the Spark packages on an Ubuntu or Debian system:

```
$ sudo apt-get install spark-core spark-master spark-worker spark-history-server
spark-python
```

You are now ready to configure Spark. See the next section.

Configuring and Running Spark (Standalone Mode)

Configuring Spark

Before you can run Spark in standalone mode, you must do the following on every host in the cluster:

- Edit the following portion of `/etc/spark/conf/spark-env.sh` to point to the host where the spark-master runs:

  ```
  ###
  ### === IMPORTANT ===
  ### Change the following to specify a real cluster's Master host
  ###
  export STANDALONE_SPARK_MASTER_HOST=`hostname`
  ```

  Change `'hostaname'` in the last line to the actual hostname of the host where the Spark master will run.

You can change other elements of the default configuration by modifying `/etc/spark/conf/spark-env.sh`. You can change the following:

- `SPARK_MASTER_PORT / SPARK_MASTER_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 1000MB, 2GB)
- `SPARK_WORKER_PORT / SPARK_WORKER_WEBUI_PORT`
- `SPARK_WORKER_INSTANCE`, to set the number of worker processes per node
- `SPARK_WORKER_DIR`, to set the working directory of worker processes

Configuring the Spark History Server

Before you can run the Spark history server, you must 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
```

On Spark clients (systems from which you intend to launch Spark jobs), do the following:

1. Create `/etc/spark/conf/spark-defaults.conf` on the Spark client:

   ```
   cp /etc/spark/conf/spark-defaults.conf.template /etc/spark/conf/spark-defaults.conf
   ```

2. Add the following to `/etc/spark/conf/spark-defaults.conf`:

   ```
   spark.eventLog.dir=/user/spark/applicationHistory
   spark.eventLog.enabled=true
   ```

This causes Spark applications running on this client to write their history to the directory that the history server reads.

In addition, if you want the YARN ResourceManager to link directly to the Spark History Server, you can set the `spark.yarn.historyServer.address` property in `/etc/spark/conf/spark-defaults.conf`:

```
spark.yarn.historyServer.address=http://HISTORY_HOST:HISTORY_PORT
```

Starting, Stopping, and Running Spark

- To start Spark, proceed as follows:
  - On one node in the cluster, start the master:

    ```
    $ sudo service spark-master start
    ```

- On all the other nodes, start the workers:

  ```
  $ sudo service spark-worker start
  ```

- On one node, start the history server:

  ```
  $ sudo service spark-history-server start
  ```

- To stop Spark, use the following commands on the appropriate hosts:

  ```
  $ sudo service spark-worker stop
  $ sudo service spark-master stop
  $ sudo service spark-history-server stop
  ```

Service logs are stored in `/var/log/spark`.

You can use the GUI for the Spark master at `<master_host>:18080`.

Testing the Spark Service

To test the Spark service, start `spark-shell` on one of the nodes. You can, for example, run a word count application:

```
val file = sc.textFile("hdfs://namenode:8020/path/to/input")
val counts = file.flatMap(line => line.split(" "))
                 .map(word => (word, 1))
                 .reduceByKey(_ + _)
counts.saveAsTextFile("hdfs://namenode:8020/output")
```

You can see the application by going to the Spark Master UI, by default at `http://spark-master:18080`, to see the Spark Shell application, its executors and logs.

Running Spark Applications

For details on running Spark applications in the YARN Client and Cluster modes, see Running Spark Applications on page 335.

Running Spark Applications

Spark applications are similar to MapReduce "jobs." Each application is a self-contained computation which runs some user-supplied code to compute a result. As with MapReduce jobs, Spark applications can make use of the resources of multiple nodes.

Each application has a driver process which coordinates its execution. This process can run in the foreground (**client mode**) or in the background (**cluster mode**). Client mode is a little simpler, but cluster mode allows you to easily log out after starting a Spark application without terminating the application.

Spark starts **executors** to perform computations. There may be many executors, distributed across the cluster, depending on the size of the job.

Spark can run in two modes:

- **Standalone mode:**

  In standalone mode, Spark uses a Master daemon which coordinates the efforts of the Workers, which run the executors. Standalone mode is the default, but it cannot be used on secure clusters.

- **YARN mode:**

  In YARN mode, the YARN ResourceManager performs the functions of the Spark Master. The functions of the Workers are performed by the YARN NodeManager daemons, which run the executors. YARN mode is slightly more complex to set up, but it supports security, and provides better integration with YARN's cluster-wide resource management policies.

Important: If you are using Cloudera Manager, the Spark assembly JAR is uploaded to HDFS automatically, as `/user/spark/share/lib/spark-assembly.jar`.

On Kerberized clusters, the upload jar command will fail silently because Spark does not have Kerberos support. To run Spark on YARN on a kerberized cluster, manually upload the Spark assembly jar to HDFS `/user/spark/share/lib`. The Spark assembly jar is located on the local filesystem, typically in `/usr/lib/spark/assembly/lib` or `/opt/cloudera/parcels/CDH/lib/spark/assembly/lib`.

Multiple Spark applications can run at once. If you decide to run Spark on YARN, you can decide on an application-by-application basis whether to run in YARN client mode or cluster mode. When you run Spark in client mode, the driver process runs locally; in cluster mode, it runs remotely on an ApplicationMaster.

The following sections use a sample application, SparkPi, which is packaged with Spark and computes the value of Pi, to illustrate the three modes.

Configuration

The easiest way to configure Spark is by setting `$SPARK_HOME/conf/spark-defaults.conf`.

This file contains lines in the form: "key value". You can create a comment by putting a hash mark (#) at the beginning of a line.

Note: You cannot add comments to the end or middle of a line.

Here is an example of a `spark-defaults.conf` file:

```
spark.master     spark://mysparkmaster.cloudera.com:7077
spark.eventLog.enabled    true
spark.eventLog.dir        hdfs:///user/spark/eventlog
# Set spark executor memory
spark.executor.memory     2g
spark.logConf             true
```

It is a good idea to put configuration keys that you want to use for every application into `spark-defaults.conf`. See Script for more information about configuration keys.

The Spark-Submit Script

You can start Spark applications with the `spark-submit` script, which is installed in your path when you install the `spark-core` package.

Note: Spark cannot handle command line options of the form `--key=value`; use `--key value` instead. (That is, use a space instead of an equals sign.)

To run `spark-submit`, you need a compiled Spark application JAR. The following sections use a sample JAR, `SparkPi`, which is packaged with Spark. It computes an approximation to the value of Pi.

Running SparkPi in Standalone Mode

Supply the `--master` and `--deploy-mode` client arguments to run `SparkPi` in standalone mode:

```
spark-submit \
--class org.apache.spark.examples.SparkPi \
--deploy-mode client \
--master spark//$SPARK_MASTER_IP:$SPARK_MASTER_PORT \
$SPARK_HOME/examples/*/scala-*/spark-examples-*.jar 10
```

Arguments that come after the JAR name are supplied to the application. In this case, the argument controls how good we want our approximation to Pi to be.

Running SparkPi in YARN Client Mode

In this case, the command to run `SparkPi` is as follows:

```
spark-submit  \
--class org.apache.spark.examples.SparkPi \
--deploy-mode client \
--master yarn \
$SPARK_HOME/examples/*/scala-*/spark-examples-*.jar 10
```

Running SparkPi in YARN Cluster Mode

In this case, the command to run `SparkPi` is a as follows:

```
spark-submit  \
--class org.apache.spark.examples.SparkPi \
--deploy-mode cluster \
--master yarn \
$SPARK_HOME/examples/*/scala-*/spark-examples-*.jar 10
```

The command will continue to print out status until the job finishes, or you press `control-C`. Terminating the `spark-submit` process in cluster mode does not terminate the Spark application as it does in client mode. To monitor the status of the running application, run `yarn application -list`.

Optimizing YARN Mode

Important: If you are using Cloudera Manager, the Spark assembly JAR is uploaded to HDFS automatically, as `/user/spark/share/lib/spark-assembly.jar`.

On Kerberized clusters, the upload jar command will fail silently because Spark does not have Kerberos support. To run Spark on YARN on a kerberized cluster, manually upload the Spark assembly jar to HDFS `/user/spark/share/lib`. The Spark assembly jar is located on the local filesystem, typically in `/usr/lib/spark/assembly/lib` or `/opt/cloudera/parcels/CDH/lib/spark/assembly/lib`.

Normally, Spark copies the Spark assembly JAR file to HDFS each time you run `spark-submit`, as you can see in the following log messages:

```

14/06/11 14:21:49 INFO yarn.Client: Uploading
file:/home/cmccabe/spark/b2.4/examples/target/scala-2.10/spark-examples-1.0.0-SNAPSHOT-hadoop2.4.0.jar
```

```
 to
hdfs://a2402.halxg.cloudera.com:6000/user/cmccabe/.sparkStaging/application_1402278226964_0012/spark-examples-1.0.0-SNAPSHOT-hadoop2.4.0.jar
14/06/11 14:21:50 INFO yarn.Client: Uploading
file:/home/cmccabe/spark/b2.4/assembly/target/scala-2.10/spark-assembly-1.0.0-SNAPSHOT-hadoop2.4.0.jar
 to
hdfs://a2402.halxg.cloudera.com:6000/user/cmccabe/.sparkStaging/application_1402278226964_0012/spark-assembly-1.0.0-SNAPSHOT-hadoop2.4.0.jar
```

You can avoid doing this copy each time by manually uploading the Spark assembly JAR file to your HDFS. Then set the `SPARK_JAR` environment variable to this HDFS path:

```
hdfs dfs -mkdir -p /user/spark/share/lib
hdfs dfs -put $SPARK_HOME/assembly/lib/spark-assembly_*.jar  \
/user/spark/share/lib/spark-assembly.jar
SPARK_JAR=hdfs://<nn>:<port>/user/spark/share/lib/spark-assembly.jar
```

Building Spark Applications

Best practices when compiling your Spark applications include:

- Building a single assembly JAR that includes all the dependencies, except those for Spark and Hadoop.
- Excluding any Spark and Hadoop classes from the assembly JAR, because they are already on the cluster, and part of the runtime classpath. In Maven, you can mark the Spark and Hadoop dependencies as `provided`.
- Always building against the same version of Spark that you are running against, to avoid compatibility issues.

  For example, do not assume that applications compiled against Spark 0.9 will run on Spark 1.0 without recompiling. In addition, some applications compiled under Spark 0.9 or earlier will need changes to their source code to compile under Spark 1.0. Applications that compile under Spark 1.0 should compile under all future versions.

Running Crunch with Spark

The blog post How-to: Run a Simple Apache Spark App in CDH 5 provides a tutorial on writing, compiling and running a Spark application. Taking that article as a starting point, do the following to run Crunch with Spark.

1. Add both the `crunch-core` and `crunch-spark` dependencies to your Maven project, along with the other dependencies shown in the blog post.
2. Use the `SparkPipeline` (`org.apache.crunch.impl.spark.SparkPipeline`) where you would have used the `MRPipeline` instance in the declaration of your Crunch pipeline. The `SparkPipeline` will need either a String that contains the connection string for the Spark master (`local` for local mode, `yarn-client` for YARN) or an actual `JavaSparkContext` instance.
3. Update the `SPARK_SUBMIT_CLASSPATH`:

   ```
   export
   SPARK_SUBMIT_CLASSPATH=./commons-codec-1.4.jar:$SPARK_HOME/assembly/lib/*:./myapp-jar-with-dependencies.jar
   ```

   Important:

   The `commons-codec-1.4` dependency must come before the `SPARK_HOME` dependencies.

4. Now you can start the pipeline using your Crunch app *jar-with-dependencies* file using the `spark-submit` script, just as you would for a regular Spark pipeline.

Sqoop 1 Installation

Apache Sqoop 1 is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. You can use Sqoop 1 to import data from external structured datastores into the Hadoop Distributed File System (HDFS) or related systems such as Hive and HBase. Conversely, you can use Sqoop 1 to extract data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses.

Note:

To see which version of Sqoop 1 is shipping in CDH 5, check the CDH Version and Packaging Information. For important information on new and changed components, see the CDH 5 Release Notes.

See the following sections for information and instructions:

- Upgrading Sqoop 1 from an Earlier CDH 5 release on page 340
- Packaging
- Prerequisites
- Installing Packages
- Installing a Tarball
- Installing the JDBC Drivers
- Setting HADOOP_MAPRED_HOME on page 343
- Apache Sqoop 1 Documentation

Also see Feature Differences - Sqoop 1 and Sqoop 2 on page 350 for major feature differences between Sqoop 1 and Sqoop 2.

Upgrading Sqoop 1 from CDH 4 to CDH 5

To upgrade Sqoop 1 from CDH 4 to CDH 5, proceed as follows.

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5 on page 57, you can skip Step 1 below and proceed with installing the new CDH 5 version of Sqoop 1.

Step 1: Remove the CDH 4 version of Sqoop 1

To remove Sqoop 1 on a Red Hat-compatible system:

```
$ sudo yum remove sqoop
```

To remove Sqoop 1 on an Ubuntu or other Debian system:

```
$ sudo apt-get remove sqoop
```

To remove Sqoop 1 on a SLES system:

```
$ sudo zypper remove sqoop
```

Step 2: Install the new version of Sqoop 1

Follow instructions under Installing the Sqoop 1 RPM Packages or Installing the Sqoop 1 Tarball.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

The upgrade is now complete.

Upgrading Sqoop 1 from an Earlier CDH 5 release

These instructions assume that you are upgrading Sqoop 1 as part of an upgrade to the latest CDH 5 release, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

To upgrade Sqoop 1 from an earlier CDH 5 release, install the new version of Sqoop 1 using one of the methods described below: Installing the Sqoop 1 RPM or Debian Packages on page 341 or .Installing the Sqoop 1 Tarball on page 341

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by `dpkg`.

Sqoop 1 Packaging

The packaging options for installing Sqoop 1 are:

- RPM packages
- Tarball
- Debian packages

Sqoop 1 Prerequisites

- Supported Operating Systems
- Oracle JDK

Installing the Sqoop 1 RPM or Debian Packages

Installing the Sqoop 1 RPM or Debian packages is more convenient than installing the Sqoop 1 tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations

The Sqoop 1 packages consist of:

- sqoop — Complete Sqoop 1 distribution
- `sqoop-metastore` — For installation of the Sqoop 1 metastore only

> **Important:**
>
> If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install Sqoop 1. For instructions, see CDH 5 Installation on page 27.

To install Sqoop 1 on a Red Hat system:

```
$ sudo yum install sqoop
```

To install Sqoop 1 on an Ubuntu or other Debian system:

```
$ sudo apt-get install sqoop
```

To install Sqoop 1 on a SLES system:

```
$ sudo zypper install sqoop
```

If you have already configured CDH on your system, there is no further configuration necessary for Sqoop 1. You can start using Sqoop 1 by using commands such as:

```
$ sqoop help
$ sqoop version
$ sqoop import
```

Installing the Sqoop 1 Tarball

The Sqoop 1 tarball is a self-contained package containing everything necessary to use Sqoop 1 with YARN on a Unix-like system.

> **Important:**
>
> Make sure you have read and understood the section on tarballs under How Packaging Affects CDH 5 Deployment on page 27 before you proceed with a tarball installation.

To install Sqoop 1 from the tarball, unpack the tarball in a convenient location. Once it is unpacked, add the `bin` directory to the shell path for easy access to Sqoop 1 commands. Documentation for users and developers can be found in the docs directory.

To install the Sqoop 1 tarball on Linux-based systems:

Run the following command:

```
$ (cd /usr/local/ && sudo tar -zxvf _<path_to_sqoop.tar.gz>_)
```

Note:

When installing Sqoop 1 from the tarball package, you must make sure that the environment variables `JAVA_HOME` and `HADOOP_MAPRED_HOME` are configured correctly. The variable `HADOOP_MAPRED_HOME` should point to the root directory of Hadoop installation. Optionally, if you intend to use any Hive or HBase related functionality, you must also make sure that they are installed and the variables `HIVE_HOME` and `HBASE_HOME` are configured correctly to point to the root directory of their respective installation.

Installing the JDBC Drivers

Sqoop 1 does not ship with third party JDBC drivers. You must download them separately and save them to the `/var/lib/sqoop/` directory. The following sections show how to install the most common JDBC Drivers.

Before you begin:

Make sure the `/var/lib/sqoop` directory exists and has the correct ownership and permissions:

```
mkdir -p /var/lib/sqoop
chown sqoop:sqoop /var/lib/sqoop
chmod 755 /var/lib/sqoop
```

This sets permissions to `drwxr-xr-x`.

Installing the MySQL JDBC Driver

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

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar
/var/lib/sqoop/
```

At the time of publication, *version* was `5.1.31`, but the version may have changed by the time you read this.

Important:

Make sure you have at least version 5.1.31. Some systems ship with an earlier version that may not work correctly with Sqoop.

Installing the Oracle JDBC Driver

You can download the JDBC Driver from the Oracle website, for example http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html. You must accept the license agreement before you can download the driver. Download the `ojdbc6.jar` file and copy it to the `/var/lib/sqoop/` directory:

```
$ sudo cp ojdbc6.jar /var/lib/sqoop/
```

Installing the Microsoft SQL Server JDBC Driver

Download the Microsoft SQL Server JDBC driver from http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774 and copy it to the /var/lib/sqoop/ directory. For example:

```
$ curl -L
'http://download.microsoft.com/download/0/2/A/02AAE597-3865-456C-AE7F-613F99F850A8/sqljdbc_4.0.2206.100_enu.tar.gz'
 | tar xz
$ sudo cp sqljdbc_4.0/enu/sqljdbc4.jar /var/lib/sqoop/
```

Installing the PostgreSQL JDBC Driver

Download the PostgreSQL JDBC driver from http://jdbc.postgresql.org/download.html and copy it to the /var/lib/sqoop/ directory. For example:

```
$ curl -L 'http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar' -o
postgresql-9.2-1002.jdbc4.jar
$ sudo cp postgresql-9.2-1002.jdbc4.jar /var/lib/sqoop/
```

Setting HADOOP_MAPRED_HOME

- For each user who will be submitting MapReduce jobs using MapReduce v2 (YARN), or running Pig, Hive, or Sqoop 1 in a YARN installation, make sure that the HADOOP_MAPRED_HOME environment variable is set correctly, as follows:

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

- For each user who will be submitting MapReduce jobs using MapReduce v1 (MRv1), or running Pig, Hive, or Sqoop 1 in an MRv1 installation, set the HADOOP_MAPRED_HOME environment variable as follows:

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

Viewing the Sqoop 1 Documentation

For additional documentation see the Sqoop 1 User Guide and the Sqoop 1 Developer Guide.

Sqoop 2 Installation

The following sections describe how to install and configure Sqoop 2:

- Upgrading Sqoop 2 from CDH 4 to CDH 5 on page 345
- About Sqoop 2
- Installing Sqoop 2
- Configuring Sqoop 2
- Starting, Stopping and Using the Server
- Apache Documentation

See also Feature Differences - Sqoop 1 and Sqoop 2 on page 350.

Upgrading Sqoop 2 from CDH 4 to CDH 5

To upgrade Sqoop 2 from CDH 4 to CDH 5, proceed as follows.

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5 on page 57, you can skip Step 1 below and proceed with installing the new CDH 5 version of Sqoop 2.

For more detailed instructions for upgrading Sqoop 2, see the Apache Sqoop Upgrade page.

Step 1: Remove the CDH 4 version of Sqoop 2

To remove Sqoop 2 on a Red Hat-compatible system:

```
$ sudo yum remove sqoop2-server sqoop2-client
```

To remove Sqoop 2 on an Ubuntu or other Debian system:

```
$ sudo apt-get remove sqoop2-server sqoop2-client
```

To remove Sqoop 2 on a SLES system:

```
$ sudo zypper remove sqoop2-server sqoop2-client
```

Step 2: Install the new version of Sqoop 2

1. Install the new version of Sqoop 2 following directions under Installing Sqoop 2 on page 347.
2. *If you have been running MRv1 on CDH 4 and will continue to run it on CDH 5:*

   a. Update `/etc/defaults/sqoop2-server` to point to MRv1:

   ```
   mv /etc/defaults/sqoop2-server.rpmnew /etc/defaults/sqoop2-server
   ```

   b. Update alternatives:

   ```
   alternatives --set sqoop2-tomcat-conf /etc/sqoop2/tomcat-conf.mr1
   ```

3. Run the upgrade tool:

```
sqoop2-tool upgrade
```

> **Important: Configuration files**
>
> - If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
> - If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

The upgrade is now complete.

Upgrading Sqoop 2 from an Earlier CDH 5 Release

These instructions assume that you are upgrading Sqoop 2 as part of an upgrade to the latest CDH 5 release, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

For more detailed instructions for upgrading Sqoop 2, see the Apache Sqoop Upgrade page.

To upgrade Sqoop 2 from an earlier CDH 5 release, proceed as follows:

1. Install the new version of Sqoop 2 following directions under Installing Sqoop 2 on page 347.
2. *If you are running MRv1 on CDH 5 Beta 1 and will continue to run it after upgrading:*
   a. Update `/etc/defaults/sqoop2-server` to point to MR1:

   ```
   mv /etc/defaults/sqoop2-server.rpmnew /etc/defaults/sqoop2-server
   ```

   b. Update alternatives:

   ```
   alternatives --set sqoop2-tomcat-conf /etc/sqoop2/tomcat-conf.mr1
   ```

3. Run the upgrade tool:

```
sqoop2-tool upgrade
```

> **Important: Configuration files**
>
> - If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
> - If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

About Sqoop 2

Sqoop 2 is a server-based tool designed to transfer data between Hadoop and relational databases. You can use Sqoop 2 to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data with Hadoop MapReduce, and then export it back into an RDBMS.

- To find out more about Sqoop 2, see Viewing the Sqoop 2 Documentation.
- To install Sqoop 2, follow the directions on this page.

Sqoop 2 Packaging

There are three packaging options for installing Sqoop 2:

- Tarball (`.tgz`) that contains both the Sqoop 2 server and the client.
- Separate RPM packages for Sqoop 2 server (`sqoop2-server`) and client (`sqoop2-client`)
- Separate Debian packages for Sqoop 2 server (`sqoop2-server`) and client (`sqoop2-client`)

Installing Sqoop 2

Sqoop 2 is distributed as two separate packages: a client package (`sqoop2-client`) and a server package (`sqoop2-server`). Install the server package on one node in the cluster; because the Sqoop 2 server acts as a MapReduce client this node must have Hadoop installed and configured.

Install the client package on each node that will act as a client. A Sqoop 2 client will always connect to the Sqoop 2 server to perform any actions, so Hadoop does not need to be installed on the client nodes.

Depending on what you are planning to install, choose the appropriate package and install it using your preferred package manager application.

Note: The Sqoop 2 packages can't be installed on the same machines as Sqoop1 packages. However you can use both versions in the same Hadoop cluster by installing Sqoop1 and Sqoop 2 on different nodes.

To install the Sqoop 2 server package on a Red-Hat-compatible system:

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install Sqoop 2. For instructions, see CDH 5 Installation on page 27.

```
$ sudo yum install sqoop2-server
```

To install the Sqoop 2 client package on an Red-Hat-compatible system:

```
$ sudo yum install sqoop2-client
```

To install the Sqoop 2 server package on a SLES system:

```
$ sudo zypper install sqoop2-server
```

To install the Sqoop 2 client package on an SLES system:

```
$ sudo zypper install sqoop2-client
```

Sqoop 2 Installation

To install the Sqoop 2 server package on an Ubuntu or Debian system:

```
$ sudo apt-get install sqoop2-server
```

To install the Sqoop 2 client package on an Ubuntu or Debian system:

```
$ sudo apt-get install sqoop2-client
```

Note:

Installing the `sqoop2-server` package creates a `sqoop-server` service configured to start Sqoop 2 at system startup time.

You are now ready to configure Sqoop 2. See the next section.

Configuring Sqoop 2

This section explains how to configure the Sqoop 2 server.

Configuring which Hadoop Version to Use

The Sqoop 2 client does not interact directly with Hadoop MapReduce, and so it does not require any MapReduce configuration.

The Sqoop 2 server can work with either MRv1 or YARN. **It cannot work with both simultaneously.**

You set the MapReduce version the Sqoop 2 server works with by means of the `alternatives` command (or `update-alternatives`, depending on your operating system):

- To use YARN:

```
alternatives --set sqoop2-tomcat-conf /etc/sqoop2/tomcat-conf.dist
```

- To use MRv1:

```
alternatives --set sqoop2-tomcat-conf /etc/sqoop2/tomcat-conf.mr1
```

Important: If you are upgrading from a release earlier than CDH 5 Beta 2

In earlier releases, the mechanism for setting the MapReduce version was the `CATALINA_BASE`variable in the `/etc/defaults/sqoop2-server` file. This does not work as of CDH 5 Beta 2, and in fact could cause problems. **Check your `/etc/defaults/sqoop2-server` file and make sure `CATALINA_BASE` is not set.**

Installing the JDBC Drivers

Sqoop 2 does not ship with third party JDBC drivers. You must download them separately and save them to the `/var/lib/sqoop2/` directory. The following sections show how to install the most common JDBC Drivers.

Installing the MySQL JDBC Driver

1. Download the MySQL JDBC driver onto the Sqoop 2 server from http://www.mysql.com/downloads/connector/j/5.1.html. You will need to sign up for an account if you don't already have one, and log in, before you can download it.

2. Copy the JAR file to the `/var/lib/sqoop2/` directory on the Sqoop 2 server. For example:

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar
/var/lib/sqoop2/
```

At the time of publication, *version* was `5.1.31`, but the version may have changed by the time you read this.

> **Important:**
>
> Make sure you have at least version 5.1.31. Some systems ship with an earlier version that may not work correctly with Sqoop.

3. Restart the Sqoop 2 server to load the driver.

Installing the Oracle JDBC Driver

1. Download the JDBC Driver from the Oracle website, for example http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html. You must accept the license agreement before you can download the driver. Download the `ojdbc6.jar` file.
2. Copy the JAR file to `/var/lib/sqoop2/` directory:

```
$ sudo cp ojdbc6.jar /var/lib/sqoop2/
```

3. Restart the Sqoop 2 server to load the driver.

Installing the Microsoft SQL Server JDBC Driver

1. Download the Microsoft SQL Server JDBC driver from http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774
2. Copy the JAR file to the `/var/lib/sqoop2/` directory. For example:

```
$ curl -L
'http://download.microsoft.com/download/0/2/A/02AAE597-3865-456C-AE7F-613F99F850A8/sqljdbc_4.0.2206.100_enu.tar.gz'
 | tar xz
$ sudo cp sqljdbc_4.0/enu/sqljdbc4.jar /var/lib/sqoop2/
```

3. Restart the Sqoop 2 server to load the driver.

Installing the PostgreSQL JDBC Driver

1. Download the PostgreSQL JDBC driver from http://jdbc.postgresql.org/download.html
2. Copy the JAR file to the `/var/lib/sqoop2/` directory. For example:

```
$ curl -L 'http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar' -o
postgresql-9.2-1002.jdbc4.jar
$ sudo cp postgresql-9.2-1002.jdbc4.jar /var/lib/sqoop2/
```

3. Restart the Sqoop 2 server to load the driver.

Starting, Stopping, and Accessing the Sqoop 2 Server

Starting the Sqoop 2 Server

After you have completed all of the required configuration steps, you can start Sqoop 2 server:

```
$ sudo /sbin/service sqoop2-server start
```

Stopping the Sqoop 2 Server

```
$ sudo /sbin/service sqoop2-server stop
```

Checking that the Sqoop 2 Server has Started

You can verify whether the server has started correctly by connecting to its HTTP interface. The simplest way is to get the server version using following command:

```
$ wget -qO - localhost:12000/sqoop/version
```

You should get a text fragment in JSON format similar to the following:

```
{"version":"1.99.2-cdh5.0.0",...}
```

Accessing the Sqoop 2 Server with the Sqoop 2 Client

Start the Sqoop 2 client:

```
sqoop2
```

Identify the host where your server is running (we will use `localhost` in this example):

```
sqoop:000> set server --host localhost
```

Test the connection by running the command `show version --all` to obtain the version number from server. You should see output similar to the following:

```
sqoop:000> show version --all
server version:
  Sqoop 1.99.2-cdh5.0.0 revision ...
  Compiled by jenkins on ...
client version:
  Sqoop 1.99.2-cdh5.0.0 revision ...
  Compiled by jenkins on ...
Protocol version:
  [1]
```

Viewing the Sqoop 2 Documentation

For more information about Sqoop 2, see Highlights_of_Sqoop 2 and http://archive.cloudera.com/cdh5/cdh/5/sqoop2.

Feature Differences - Sqoop 1 and Sqoop 2

Note:

Moving from Sqoop 1 to Sqoop 2: Sqoop 2 is essentially the future of the Apache Sqoop project. However, since Sqoop 2 currently lacks some of the features of Sqoop 1, Cloudera recommends you use Sqoop 2 only if it contains all the features required for your use case, otherwise, continue to use Sqoop 1.

Feature	Sqoop 1	Sqoop 2
Connectors for all major RDBMS	Supported.	Not supported. **Workaround**: Use the generic JDBC Connector which has been tested on the following databases: Microsoft SQL Server, PostgreSQL, MySQL and Oracle. This connector should work on any other JDBC compliant database. However, performance might not be comparable to that of specialized connectors in Sqoop.
Kerberos Security Integration	Supported.	Not supported.
Encryption of Stored Passwords	Not supported. No workaround.	Supported using Derby's on-disk encryption. **Disclaimer:** Although expected to work in the current version of Sqoop 2, this configuration has not been verified.
Data transfer from RDBMS to Hive or HBase	Supported.	Not supported. **Workaround:** Follow this two-step approach. **1.** Import data from RDBMS into HDFS **2.** Load data into Hive or HBase manually using appropriate tools and commands such as the `LOAD DATA` statement in Hive
Data transfer from Hive or HBase to RDBMS	Not supported. **Workaround:** Follow this two-step approach. **1.** Extract data from Hive or HBase into HDFS (either as a text or Avro file) **2.** Use Sqoop to export output of previous step to RDBMS	Not supported. Follow the same workaround as for Sqoop 1.

Whirr Installation

Apache Whirr is a set of libraries for running cloud services. You can use Whirr to run CDH 5 clusters on cloud providers' clusters, such as Amazon Elastic Compute Cloud (Amazon EC2). There's no need to install the RPMs for CDH 5 or do any configuration; a working cluster will start immediately with one command. It's ideal for running temporary Hadoop clusters to carry out a proof of concept, or to run a few one-time jobs. When you are finished, you can destroy the cluster and all of its data with one command.

Important:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install or update Whirr. For instructions, see CDH 5 Installation on page 27.

Use the following sections to install, upgrade, and deploy Whirr:

- Upgrading Whirr
- Installing Whirr
- Generating an SSH Key Pair
- Defining a Cluster
- Launching a Cluster
- Apache Whirr Documentation

Upgrading Whirr

Note:

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

Upgrading Whirr from CDH 4 to CDH 5

To upgrade Whirr from CDH 4, uninstall the CDH 4 version, modify the properties file, and install the CDH 5 version. Proceed as follows.

Note:

If you have already performed the steps to uninstall CDH 4 and all components, as described under Upgrading from CDH 4 to CDH 5, you can skip Step 1 below and proceed with installing the new CDH 5 version of Whirr.

Step 1: Remove Whirr

1. Stop the Whirr proxy. Kill the `hadoop-proxy.sh` process by pressing Control-C.
2. Destroy the Cluster. Whirr clusters are normally short-lived. If you have a running cluster, destroy it: see Destroying a cluster.
3. Uninstall the CDH 4 version of Whirr:

   On Red Hat-compatible systems:

```
$ sudo yum remove whirr
```

On SLES systems:

```
$ sudo zypper remove whirr
```

On Ubuntu and Debian systems:

```
sudo apt-get remove whirr
```

4. Update the Properties File.

   Edit the configuration file, called `hadoop.properties` in these instructions, and save it.

   - For Hadoop, configure the following properties as shown:
     - For MRv1:

       ```
       whirr.env.repo=cdh5
       whirr.hadoop.install-function=install_cdh_hadoop
       whirr.hadoop.configure-function=configure_cdh_hadoop
       ```

     - For YARN: see Defining a Cluster.
   - For HBase, configure the following properties as shown:

     ```
     whirr.env.repo=cdh5
     whirr.hadoop.install-function=install_cdh_hadoop
     whirr.hadoop.configure-function=configure_cdh_hadoop
     whirr.hbase.install-function=install_cdh_hbase
     whirr.hbase.configure-function=configure_cdh_hbase
     whirr.zookeeper.install-function=install_cdh_zookeeper
     whirr.zookeeper.configure-function=configure_cdh_zookeeper
     ```

   - For ZooKeeper, configure the following properties as shown:

     ```
     whirr.env.repo=cdh5
     whirr.zookeeper.install-function=install_cdh_zookeeper
     whirr.zookeeper.configure-function=configure_cdh_zookeeper
     ```

   See Defining a Whirr Cluster for a sample file.

   Important:

   If you are upgrading from Whirr version 0.3.0, and are using an explicit image (AMI), make sure it comes from one of the supplied Whirr recipe files.

Step 2: Install the new Version

See the next section, Installing Whirr.

The upgrade is now complete. For more information, see Defining a Whirr Cluster, Launching a Cluster, and Viewing the Whirr Documentation.

Upgrading Whirr from an Earlier CDH 5 Release to the Latest CDH 5 Release

Step 1: Stop the Whirr proxy.

Kill the `hadoop-proxy.sh` process by pressing Control-C.

Step 2: Destroy the Cluster.

Whirr clusters are normally short-lived. If you have a running cluster, destroy it: see Destroying a cluster on page 358.

Step 3: Install the New Version of Whirr

See Installing Whirr on page 355.

The upgrade is now complete. For more information, see Launching a Cluster on page 356 ,and Viewing the Whirr Documentation on page 358.

Installing Whirr

To install Whirr on an Ubuntu or other Debian system:

```
$ sudo apt-get install whirr
```

To install Whirr on a Red Hat system:

```
$ sudo yum install whirr
```

To install Whirr on a SLES system:

```
$ sudo zypper install whirr
```

To install Whirr on another system: Download a Whirr tarball from here.

To verify Whirr is properly installed:

```
$ whirr version
```

Generating an SSH Key Pair

After installing Whirr, generate a password-less SSH key pair to enable secure communication with the Whirr cluster.

```
ssh-keygen -t rsa -P ''
```

Note:

If you specify a non-standard location for the key files in the `ssh-keygen` command (that is, not ~/.ssh/id_rsa), then you must specify the location of the private key file in the `whirr.private-key-file` property and the public key file in the `whirr.public-key-file` property. For more information, see the next section.

Defining a Whirr Cluster

Note:

For information on finding your cloud credentials, see the Whirr FAQ.

After generating an SSH key pair, the only task left to do before using Whirr is to define a cluster by creating a properties file. You can name the properties file whatever you like. The example properties file used in these instructions is named `hadoop.properties`. Save the properties file in your home directory. After defining a cluster in the properties file, you will be ready to launch a cluster and run MapReduce jobs.

Important:

The properties shown below are sufficient to get a bare-bones cluster up and running, but you will probably need to do more configuration to do real-life tasks, especially if you are using HBase and ZooKeeper. You can find more comprehensive template files in the `recipes` directory, for example `recipes/hbase-cdh.properties`.

MRv1 Cluster

The following file defines a cluster with a single machine for the NameNode and JobTracker, and another machine for a DataNode and TaskTracker.

```
whirr.cluster-name=myhadoopcluster
whirr.instance-templates=1 hadoop-jobtracker+hadoop-namenode,1
hadoop-datanode+hadoop-tasktracker
whirr.provider=aws-ec2
whirr.identity=<cloud-provider-identity>
whirr.credential=<cloud-provider-credential>
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa.pub
whirr.env.repo=cdh5
whirr.hadoop-install-function=install_cdh_hadoop
whirr.hadoop-configure-function=configure_cdh_hadoop
whirr.hardware-id=m1.large
whirr.image-id=us-east-1/ami-ccb35ea5
whirr.location-id=us-east-1
```

YARN Cluster

The following configuration provides the essentials for a YARN cluster. Change the number of instances for `hadoop-datanode+yarn-nodemanager` from 2 to a larger number if you need to.

```
whirr.cluster-name=myhadoopcluster
whirr.instance-templates=1 hadoop-namenode+yarn-resourcemanager+mapreduce-historyserver,2
 hadoop-datanode+yarn-nodemanager
whirr.provider=aws-ec2
whirr.identity=<cloud-provider-identity>
whirr.credential=<cloud-provider-credential>
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa.pub
whirr.env.mapreduce_version=2
whirr.env.repo=cdh5
whirr.hadoop.install-function=install_cdh_hadoop
whirr.hadoop.configure-function=configure_cdh_hadoop
whirr.mr_jobhistory.start-function=start_cdh_mr_jobhistory
whirr.yarn.configure-function=configure_cdh_yarn
whirr.yarn.start-function=start_cdh_yarn
whirr.hardware-id=m1.large
whirr.image-id=us-east-1/ami-ccb35ea5
whirr.location-id=us-east-1
```

Launching a Cluster

To launch a cluster:

```
$ whirr launch-cluster --config hadoop.properties
```

As the cluster starts up, messages are displayed in the console. You can see debug-level log messages in a file named `whirr.log` in the directory where you ran the `whirr` command. After the cluster has started, a message appears in the console showing the URL you can use to access the web UI for Whirr.

Running a Whirr Proxy

For security reasons, traffic from the network where your client is running is proxied through the master node of the cluster using an SSH tunnel (a SOCKS proxy on port 6666). A script to launch the proxy is created when you launch the cluster, and may be found in `~/.whirr/<cluster-name>`.

To launch the Whirr proxy:

1. Run the following command in a new terminal window:

```
$ . ~/.whirr/myhadoopcluster/hadoop-proxy.sh
```

2. To stop the proxy, kill the process by pressing Ctrl-C.

Running a MapReduce job

After you launch a cluster, a `hadoop-site.xml` file is automatically created in the directory `~/.whirr/<cluster-name>`. You need to update the local Hadoop configuration to use this file.

To update the local Hadoop configuration to use hadoop-site.xml:

1. On all systems, type the following commands:

```
$ cp -r /etc/hadoop/conf.empty /etc/hadoop/conf.whirr
$ rm -f /etc/hadoop/conf.whirr/*-site.xml
$ cp ~/.whirr/myhadoopcluster/hadoop-site.xml /etc/hadoop/conf.whirr
```

2. If you are using an Ubuntu, Debian, or SLES system, type these commands:

```
$ sudo update-alternatives --install /etc/hadoop/conf hadoop-conf
/etc/hadoop/conf.whirr 50
$ update-alternatives --display hadoop-conf
```

3. If you are using a Red Hat system, type these commands:

```
$ sudo alternatives --install /etc/hadoop/conf hadoop-conf /etc/hadoop/conf.whirr
 50
$ alternatives --display hadoop-conf
```

4. You can now browse HDFS:

```
$ hadoop fs -ls /
```

To run a MapReduce job, run these commands:

- For MRv1:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-0.20-mapreduce
$ hadoop fs -mkdir input
$ hadoop fs -put $HADOOP_MAPRED_HOME/CHANGES.txt input
$ hadoop jar $HADOOP_MAPRED_HOME/hadoop-examples.jar wordcount input output
$ hadoop fs -cat output/part-* | head
```

- For YARN:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
$ hadoop fs -mkdir input
$ hadoop fs -put $HADOOP_MAPRED_HOME/CHANGES.txt input
$ hadoop jar $HADOOP_MAPRED_HOME/hadoop-mapreduce-examples.jar wordcount input output
$ hadoop fs -cat output/part-* | head
```

Destroying a cluster

When you are finished using a cluster, you can terminate the instances and clean up the resources using the commands shown in this section.

WARNING

All data will be deleted when you destroy the cluster.

To destroy a cluster:

1. Run the following command to destroy a cluster:

```
$ whirr destroy-cluster --config hadoop.properties
```

2. Shut down the SSH proxy to the cluster if you started one earlier.

Viewing the Whirr Documentation

For additional documentation see the Whirr Documentation.

ZooKeeper Installation

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to / and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

Apache ZooKeeper is a highly reliable and available service that provides coordination between distributed processes.

Note: For More Information

From the Apache ZooKeeper site:

> ZooKeeper is a high-performance coordination service for distributed applications. It exposes common services — such as naming, configuration management, synchronization, and group services - in a simple interface so you don't have to write them from scratch. You can use it off-the-shelf to implement consensus, group management, leader election, and presence protocols. And you can build on it for your own, specific needs.

To learn more about Apache ZooKeeper, visit http://zookeeper.apache.org/.

Note:

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

Use the following sections to install, upgrade and administer ZooKeeper:

- Upgrading ZooKeeper from CDH 4 to CDH 5 on page 359
- Upgrading ZooKeeper from an Earlier CDH 5 Release on page 360
- Installing the ZooKeeper Packages on page 361
- Maintaining a ZooKeeper Server on page 364
- Viewing the ZooKeeper Documentation on page 364

Upgrading ZooKeeper from CDH 4 to CDH 5

To upgrade ZooKeeper from CDH 4 to CDH 5, uninstall the CDH 4 version (if you have not already done so) and then install the CDH 5 version. Do the following on each server.

Note: If you have already performed the steps to uninstall CDH 4 described under Upgrading from CDH 4 to CDH 5, you can skip Step 1 below and proceed with Step 2.

ZooKeeper Installation

Step 1: Remove ZooKeeper

1. Stop the ZooKeeper server:

   ```
   $ sudo service zookeeper-server stop
   ```

   or

   ```
   $ sudo service zookeeper stop
   ```

   depending on the platform and release.

2. Remove CDH 4 ZooKeeper

   To remove ZooKeeper on Red Hat-compatible systems:

   ```
   $ sudo yum remove zookeeper-server
   ```

   To remove ZooKeeper on Ubuntu and Debian systems:

   ```
   $ sudo apt-get remove zookeeper-server
   ```

   To remove ZooKeeper on SLES systems:

   ```
   $ sudo zypper remove zookeeper-server
   ```

Step 2: Install the ZooKeeper Base Package

See Installing the ZooKeeper Base Package.

Step 3: Install the ZooKeeper Server Package

See Installing the ZooKeeper Server Package.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Step 5: Restart the Server

See Installing the ZooKeeper Server Package for instructions on starting the server.

Upgrading ZooKeeper from an Earlier CDH 5 Release

Cloudera recommends that you use a **rolling upgrade** process to upgrade ZooKeeper: that is, upgrade one server in the ZooKeeper ensemble at a time. This means bringing down each server in turn, upgrading the software, then restarting the server. The server will automatically rejoin the quorum, update its internal state with the current ZooKeeper leader, and begin serving client sessions.

This method allows you to upgrade ZooKeeper without any interruption in the service, and also lets you monitor the ensemble as the upgrade progresses, and roll back if necessary if you run into problems.

The instructions that follow assume that you are upgrading ZooKeeper as part of a CDH 5 upgrade, and have already performed the steps under Upgrading from a CDH 5 Beta Release to the Latest Version on page 76.

Performing a ZooKeeper Rolling Upgrade

Follow these steps to perform a rolling upgrade.

Step 1: Stop the ZooKeeper Server on the First Node

To stop the ZooKeeper server:

```
$ sudo service zookeeper-server stop
```

Step 2: Install the ZooKeeper Base Package on the First Node

See Installing the ZooKeeper Base Package.

Step 3: Install the ZooKeeper Server Package on the First Node

See Installing the ZooKeeper Server Package.

Important: Configuration files

- If you install a newer version of a package that is already on the system, configuration files that you have modified will remain intact.
- If you uninstall a package, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. If you then re-install the package (probably to install a new version) the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see Automatic handling of configuration files by dpkg.

Step 4: Restart the Server

See Installing the ZooKeeper Server Package for instructions on starting the server.

The upgrade is now complete on this server and you can proceed to the next.

Step 5: Upgrade the Remaining Nodes

Repeat Steps 1-4 above on each of the remaining nodes.

The ZooKeeper upgrade is now complete.

Installing the ZooKeeper Packages

There are two ZooKeeper server packages:

- The `zookeeper` base package provides the basic libraries and scripts that are necessary to run ZooKeeper servers and clients. The documentation is also included in this package.
- The `zookeeper-server` package contains the `init.d` scripts necessary to run ZooKeeper as a daemon process. Because `zookeeper-server` depends on `zookeeper`, installing the server package automatically installs the base package.

Note:

If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` repository before using the following commands to install ZooKeeper. For instructions, see CDH 5 Installation on page 27.

Installing the ZooKeeper Base Package

To install ZooKeeper On Red Hat-compatible systems:

```
$ sudo yum install zookeeper
```

To install ZooKeeper on Ubuntu and other Debian systems:

```
$ sudo apt-get install zookeeper
```

To install ZooKeeper on SLES systems:

```
$ sudo zypper install zookeeper
```

Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server

The instructions provided here deploy a single ZooKeeper server in "standalone" mode. This is appropriate for evaluation, testing and development purposes, but may not provide sufficient reliability for a production application. See Installing ZooKeeper in a Production Environment on page 363 for more information.

To install the ZooKeeper Server On Red Hat-compatible systems:

```
$ sudo yum install zookeeper-server
```

To install a ZooKeeper server on Ubuntu and other Debian systems:

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

To install ZooKeeper on SLES systems:

```
$ sudo zypper install zookeeper-server
```

To create `/var/lib/zookeeper` and set permissions:

```
mkdir -p /var/lib/zookeeper
chown -R zookeeper /var/lib/zookeeper/
```

To start ZooKeeper

Note:

ZooKeeper may start automatically on installation on Ubuntu and other Debian systems. This automatic start will happen only if the data directory exists; otherwise you will be prompted to initialize as shown below.

- **To start ZooKeeper after an upgrade:**

```
$ sudo service zookeeper-server start
```

- **To start ZooKeeper after a first-time install:**

```
$ sudo service zookeeper-server init
$ sudo service zookeeper-server start
```

> **Note:**
>
> If you are deploying multiple ZooKeeper servers after a fresh install, you need to create a `myid` file in the data directory. You can do this by means of an `init` command option: `$ sudo service zookeeper-server init --myid=1`

Installing ZooKeeper in a Production Environment

In a production environment, you should deploy ZooKeeper as an ensemble with an odd number of nodes. 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 it is recommended that each server run on a separate machine.

Deploying a ZooKeeper ensemble requires some additional configuration. The configuration file (`zoo.cfg`) on each server must include a list of all servers in the ensemble, and each server must also have a `myid` file in its data directory (by default `/var/lib/zookeeper`) that identifies it as one of the servers in the ensemble. Proceed as follows *on each server*.

1. Use the commands under Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server on page 362 to install `zookeeper-server` on each host.
2. Test the expected loads to set the Java heap size so as to avoid swapping. Make sure you are well below the threshold at which the system would start swapping; for example 12GB for a machine with 16GB of RAM.
3. Create a configuration file. This file can be called anything you like, and must specify settings for at least the parameters shown under "Minimum Configuration" in the ZooKeeper Administrator's Guide. You should also configure values for `initLimit`, `sycLimit`, and `server.`*n*; see the explanations in the administrator's guide. For example:

   ```
   tickTime=2000
   dataDir=/var/lib/zookeeper/
   clientPort=2181
   initLimit=5
   syncLimit=2
   server.1=zoo1:2888:3888
   server.2=zoo2:2888:3888
   server.3=zoo3:2888:3888
   ```

   In this example, the final three lines are in the form `server.`*`id=hostname:port:port`*. The first port is for a follower in the ensemble to listen on for the leader; the second is for leader election. You set *id* for each server in the next step.

4. Create a file named `myid` in the server's `DataDir`; in this example, `/var/lib/zookeeper/myid` . The file must contain only a single line, and that line must consist of a single unique number between 1 and 255; this is the *id* component mentioned in the previous step. In this example, the server whose hostname is `zoo1` must have a `myid` file that contains only `1`.
5. Start each server as described in the previous section.
6. Test the deployment by running a ZooKeeper client:

   ```
   zookeeper-client -server hostname:port
   ```

   For example:

   ```
   zookeeper-client -server zoo1:2181
   ```

For more information on configuring a multi-server deployment, see Clustered (Multi-Server) Setup in the ZooKeeper Administrator's Guide.

Setting up Supervisory Process for the ZooKeeper Server

The ZooKeeper server is designed to be both highly reliable and highly available. This means that:

- If a ZooKeeper server encounters an error it cannot recover from, it will "fail fast" (the process will exit immediately)
- When the server shuts down, the ensemble remains active, and continues serving requests
- Once restarted, the server rejoins the ensemble without any further manual intervention.

Cloudera recommends that you fully automate this process by configuring a supervisory service to manage each server, and restart the ZooKeeper server process automatically if it fails. See the ZooKeeper Administrator's Guide for more information.

Maintaining a ZooKeeper Server

The ZooKeeper server continually saves znode snapshot files and, optionally, transactional logs in a Data Directory to enable you to recover data. It's a good idea to back up the ZooKeeper Data Directory periodically. Although ZooKeeper is highly reliable because a persistent copy is replicated on each server, recovering from backups may be necessary if a catastrophic failure or user error occurs.

When you use the default configuration, the ZooKeeper server does not remove the snapshots and log files, so they will accumulate over time. You will need to clean up this directory occasionally, taking into account on your backup schedules and processes. To automate the cleanup, a `zkCleanup.sh` script is provided in the `bin` directory of the `zookeeper` base package. Modify this script as necessary for your situation. In general, you want to run this as a `cron` task based on your backup schedule.

The data directory is specified by the `dataDir` parameter in the ZooKeeper configuration file, and the data log directory is specified by the `dataLogDir` parameter.

For more information, see Ongoing Data Directory Cleanup.

Viewing the ZooKeeper Documentation

For additional ZooKeeper documentation, see http://archive.cloudera.com/cdh5/cdh/5/zookeeper/.

Avro Usage

Apache Avro is a serialization system. Avro supports rich data structures, a compact binary encoding, and a container file for sequences of Avro data (often referred to as "Avro data files"). Avro is designed to be language-independent and there are several language bindings for it, including Java, C, C++, Python, and Ruby.

Avro does not rely on generated code, which means that processing data imported from Flume or Sqoop 1 is simpler than using Hadoop Writables in Sequence Files, where you have to take care that the generated classes are on the processing job's classpath. Furthermore, Pig and Hive cannot easily process Sequence Files with custom Writables, so users often revert to using text, which has disadvantages from a compactness and compressibility point of view (compressed text is not generally splittable, making it difficult to process efficiently using MapReduce).

All components in CDH 5 that produce or consume files support Avro data files as a file format. But bear in mind that because uniform Avro support is new, there may be some rough edges or missing features.

The following sections contain brief notes on how to get started using Avro in the various CDH 5 components:

- Avro Data Files
- Compression
- Flume
- Sqoop
- MapReduce
- Streaming
- Pig
- Hive
- Avro Tools

Avro Data Files

Avro data files have the `.avro` extension. Make sure the files you create have this extension, since some tools look for it to determine which files to process as Avro (e.g. `AvroInputFormat` and `AvroAsTextInputFormat` for MapReduce and Streaming).

Compression

By default Avro data files are not compressed, but it is generally advisable to enable compression to reduce disk usage and increase read and write performance. Avro data files support Deflate and Snappy compression. Snappy is faster, while Deflate is slightly more compact.

You do not need to do any additional configuration to read a compressed Avro data file rather than an uncompressed one. However, to write an Avro data file you need to specify the type of compression to use. How you specify compression depends on the component being used, as explained in the sections below.

Flume

The HDFSEventSink that is used to serialize event data onto HDFS supports plugin implementations of EventSerializer interface. Implementations of this interface have full control over the serialization format and can be used in cases where the default serialization format provided by the Sink does not suffice.

An abstract implementation of the EventSerializer interface is provided along with Flume, called the AbstractAvroEventSerializer. This class can be extended to support custom schema for Avro serialization over

HDFS. A simple implementation that maps the events to a representation of String header map and byte payload in Avro is provided by the class FlumeEventAvroEventSerializer which can be used by setting the serializer property of the Sink as follows:

<agent-name>.sinks.<sink-name>.serializer = AVRO_EVENT

Sqoop 1

On the command line, use the following option to import to Avro data files:

```
--as-avrodatafile
```

Sqoop 1 will automatically generate an Avro schema that corresponds to the database table being exported from.

To enable Snappy compression, add the following option:

```
--compression-codec snappy
```

Note:

Sqoop 2 does not currently support Avro.

MapReduce

The Avro MapReduce API is an Avro module for running MapReduce programs which produce or consume Avro data files.

If you are using Maven, simply add the following dependency to your POM:

```
<dependency>
    <groupId>org.apache.avro</groupId>
    <artifactId>avro-mapred</artifactId>
    <version>1.7.3</version>
    <classifier>hadoop2</classifier>
</dependency>
```

Then write your program using the Avro MapReduce javadoc for guidance.

At runtime, include the `avro` and `avro-mapred` JARs in the `HADOOP_CLASSPATH`; and the `avro`, `avro-mapred` and `paranamer` JARs in `-libjars`.

To enable Snappy compression on output files call `AvroJob.setOutputCodec(job, "snappy")` when configuring the job. You will also need to include the `snappy-java` JAR in `-libjars`.

Streaming

To read from Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroAsTextInputFormat` as the input format. This input format will convert each datum in the Avro data file to a string. For a `"bytes"` schema, this will be the raw bytes, while in the general case it will be a single-line JSON representation of the datum.

To write to Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroTextOutputFormat` as the output format. This output format will create Avro data files with a `"bytes"` schema, where each datum is a tab-delimited key-value pair.

At runtime specify the `avro`, `avro-mapred` and `paranamer` JARs in `-libjars` in the streaming command.

To enable Snappy compression on output files, set the property `avro.output.codec` to snappy. You will also need to include the `snappy-java` JAR in `-libjars`.

Pig

CDH provides `AvroStorage` for Avro integration in Pig.

To use it, first register the `piggybank` JAR file and supporting libraries:

```
REGISTER piggybank.jar
REGISTER lib/avro-1.7.3.jar
REGISTER lib/json-simple-1.1.jar
REGISTER lib/snappy-java-1.0.4.1.jar
```

Then you can load Avro data files as follows:

```
a = LOAD 'my_file.avro' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

Pig maps the Avro schema to a corresponding Pig schema.

You can store data in Avro data files with:

```
store b into 'output' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

In the case of `store`, Pig generates an Avro schema from the Pig schema. It is possible to override the Avro schema, either by specifying it literally as a parameter to `AvroStorage`, or by using the same schema as an existing Avro data file. See the Pig wiki for details.

To store two relations in one script, specify an index to each `store` function. Here is an example:

```
set1 = load 'input1.txt' using PigStorage() as ( ... );
store set1 into 'set1' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
 '1');

set2 = load 'input2.txt' using PigStorage() as ( ... );
store set2 into 'set2' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
 '2');
```

For more information, see the AvroStorage wiki; look for "index".

To enable Snappy compression on output files do the following before issuing the `STORE` statement:

```
SET mapred.output.compress true
SET mapred.output.compression.codec org.apache.hadoop.io.compress.SnappyCodec
SET avro.output.codec snappy
```

There is some additional documentation on the Pig wiki. Note, however, that the version numbers of the JAR files to register are different on that page, so you should adjust them as shown above.

Hive

The following example demonstrates how to create a Hive table that is backed by Avro data files:

```
CREATE TABLE doctors
ROW FORMAT
SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat'
TBLPROPERTIES ('avro.schema.literal'='{
  "namespace": "testing.hive.avro.serde",
  "name": "doctors",
  "type": "record",
  "fields": [
    {
      "name":"number",
      "type":"int",
      "doc":"Order of playing the role"
    },
    {
      "name":"first_name",
      "type":"string",
      "doc":"first name of actor playing role"
    },
    {
      "name":"last_name",
      "type":"string",
      "doc":"last name of actor playing role"
    },
    {
      "name":"extra_field",
      "type":"string",
      "doc:":"an extra field not in the original file",
      "default":"fishfingers and custard"
    }
  ]
}');

LOAD DATA LOCAL INPATH '/usr/share/doc/hive-0.7.1+42.55/examples/files/doctors.avro'
INTO TABLE doctors;
```

You could also create a Avro backed Hive table by using an Avro schema file:

```
CREATE TABLE my_avro_table(notused INT)
  ROW FORMAT SERDE
  'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
  WITH SERDEPROPERTIES (
    'avro.schema.url'='file:///tmp/schema.avsc')
  STORED as INPUTFORMAT
  'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
  OUTPUTFORMAT
  'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat';
```

The `avro.schema.url` is a URL (here a `file://` URL) pointing to an Avro schema file that is used for reading and writing, it could also be an hdfs URL, eg. `hdfs://hadoop-namenode-uri/examplefile`

To enable Snappy compression on output files, run the following before writing to the table:

```
SET hive.exec.compress.output=true;
SET avro.output.codec=snappy;
```

You will also need to include the `snappy-java` JAR in `--auxpath`. The `snappy-java` JAR is located at:

```
/usr/lib/hive/lib/snappy-java-1.0.4.1.jar
```

Haivvreo SerDe has been merged into Hive as AvroSerDe, and it is no longer supported in its original form. `schema.url` and `schema.literal` have been changed to `avro.schema.url` and `avro.schema.literal` as a result of the merge. If you were you using Haivvreo SerDe, you can use the new Hive AvroSerDe with tables created with the Haivvreo SerDe. For example, if you have a table `my_avro_table` that uses the Haivvreo SerDe, you can do the following to make the table use the new AvroSerDe:

```
ALTER TABLE my_avro_table SET SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe';

ALTER TABLE my_avro_table SET FILEFORMAT
INPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat';
```

Avro Tools

Avro provides a set of tools for working with Avro data files and schemas. The tools are not (currently) packaged with CDH, but you can download the tools JAR from an Apache mirror, and run it as follows to get a list of commands:

```
java -jar avro-tools-1.7.3.jar
```

See also RecordBreaker for information on turning text data into structured Avro data.

Using the Parquet File Format with Impala, Hive, Pig, and MapReduce

Parquet is automatically installed when you install any of the above components, and the necessary libraries are automatically placed in the classpath for all of them. Copies of the libraries are in `/usr/lib/parquet` or inside the parcels in `/lib/parquet`.

The Parquet file format incorporates several features that make it highly suited to data warehouse-style operations:

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

Among components of the CDH distribution, Parquet support originated in Cloudera Impala. Impala can create Parquet tables, insert data into them, convert data from other file formats to Parquet, and then perform SQL queries on the resulting data files. Parquet tables created by Impala can be accessed by Hive, and vice versa.

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

Using Parquet Tables with Impala

The Cloudera Impala component can create tables that use Parquet data files; insert data into those tables, converting the data into Parquet format; and query Parquet data files produced by Impala or by other components. The only syntax required is the `STORED AS PARQUET` clause on the `CREATE TABLE` statement. After that, all `SELECT`, `INSERT`, and other statements recognize the Parquet format automatically. For example, a session in the `impala-shell` interpreter might look as follows:

```
[localhost:21000] > create table parquet_table (x int, y string) stored as parquet;
[localhost:21000] > insert into parquet_table select x, y from some_other_table;
Inserted 50000000 rows in 33.52s
[localhost:21000] > select y from parquet_table where x between 70 and 100;
```

Once you create a Parquet table this way in Impala, you can query it or insert into it through either Impala or Hive.

Remember that Parquet format is optimized for working with large data files, typically 1 GB each. Avoid using the `INSERT ... VALUES` syntax, or partitioning the table at too granular a level, if that would produce a large number of small files that cannot take advantage of the Parquet optimizations for large data chunks.

Inserting data into a partitioned Impala table can be a memory-intensive operation, because each data file requires a 1 GB memory buffer to hold the data before being written. Such inserts can also exceed HDFS limits on simultaneous open files, because each node could potentially write to a separate data file for each partition, all at the same time. Consider splitting up such insert operations into one `INSERT` statement per partition.

For complete instructions and examples, see the Parquet section in the Impala documentation.

Using Parquet Tables in Hive

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

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

Note:

- Once you create a Parquet table this way in Hive, you can query it or insert into it through either Impala or Hive. Before the first time you access a newly created Hive table through Impala, issue a one-time `INVALIDATE METADATA` statement in the `impala-shell` interpreter to make Impala aware of the new table.
- `dfs.block.size` should be set to 1GB in `hdfs-site.xml`.

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

```
hive> create external table parquet_table_name (x INT, y STRING)
  ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
  STORED AS
    INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
    OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat"
    LOCATION '/test-warehouse/tinytable';
```

To populate the table with an `INSERT` statement, and to read the table with a `SELECT` statement, see the Impala documentation for Parquet.

Select the compression to use when writing data with the `parquet.compression` property, for example:

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

The valid options for compression are:

- `UNCOMPRESSED`
- `GZIP`
- `SNAPPY`

Using Parquet Files in Pig

Reading Parquet Files in Pig

Assuming the external table was created and populated with Impala or Hive as described above, the Pig instruction to read the data is:

```
grunt> A = LOAD '/test-warehouse/tinytable' USING ParquetLoader AS (x: int, y  int);
```

Writing Parquet Files in Pig

Create and populate a Parquet file with the `ParquetStorer` class:

```
grunt> store A into '/test-warehouse/tinytable' USING ParquetStorer;
```

There are three compression options: `uncompressed`, `snappy`, and `gzip`. The default is snappy. You can specify one of them once before the first store instruction in a Pig script:

```
SET parquet.compression gzip;
```

Using Parquet Files in MapReduce

MapReduce needs thrift in its `CLASSPATH` and in `libjars` to access Parquet files. It also needs `parquet-format` in `libjars`. Perform the following setup before running MapReduce jobs that access Parquet data files:

```
if [ -e /opt/cloudera/parcels/CDH ] ; then
    CDH_BASE=/opt/cloudera/parcels/CDH
else
    CDH_BASE=/usr
fi
THRIFTJAR=`ls -l $CDH_BASE/lib/hive/lib/libthrift*jar | awk '{print $9}' | head -1`
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$THRIFTJAR
export LIBJARS=`echo "$CLASSPATH" | awk 'BEGIN { RS = ":" } { print }' | grep
parquet-format | tail -1`
export LIBJARS=$LIBJARS,$THRIFTJAR

hadoop jar my-parquet-mr.jar -libjars $LIBJARS
```

Reading Parquet Files in MapReduce

Taking advantage of the `Example` helper classes in the Parquet JAR files, a simple map-only MapReduce job that reads Parquet files can use the `ExampleInputFormat` class and the `Group` value class. There is nothing special about the reduce phase when using Parquet files. The following example demonstrates how to read a Parquet file in a MapReduce job; portions of code specific to the Parquet aspect are shown in bold.

```
import static java.lang.Thread.sleep;
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.ExampleInputFormat;

public class TestReadParquet extends Configured
  implements Tool {
  private static final Log LOG =
  Log.getLog(TestReadParquet.class);

    /*
     * Read a Parquet record
     */
    public static class MyMap extends
      Mapper<LongWritable, Group, NullWritable, Text> {

      @Override
     public void map(LongWritable key, Group value, Context context) throws IOException,
 InterruptedException {
          NullWritable outKey = NullWritable.get();
          String outputRecord = "";
          // Get the schema and field values of the record
          String inputRecord = value.toString();
          // Process the value, create an output record
          // ...
          context.write(outKey, new Text(outputRecord));
      }
  }

  public int run(String[] args) throws Exception {

    Job job = new Job(getConf());

    job.setJarByClass(getClass());
    job.setJobName(getClass().getName());
    job.setMapOutputKeyClass(LongWritable.class);
    job.setMapOutputValueClass(Text.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(Text.class);
    job.setMapperClass(MyMap.class);
    job.setNumReduceTasks(0);

    job.setInputFormatClass(ExampleInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);

    FileInputFormat.setInputPaths(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.waitForCompletion(true);
    return 0;
  }

  public static void main(String[] args) throws Exception {
    try {
      int res = ToolRunner.run(new Configuration(), new TestReadParquet(), args);
      System.exit(res);
    } catch (Exception e) {
      e.printStackTrace();
      System.exit(255);
    }
  }
}
```

Writing Parquet Files in MapReduce

When writing Parquet files you will need to provide a schema. The schema can be specified in the run method of the job before submitting it, for example:

```
...
import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.GroupWriteSupport;
import parquet.hadoop.example.ExampleInputFormat;
import parquet.hadoop.example.ExampleOutputFormat;
import parquet.hadoop.metadata.CompressionCodecName;
import parquet.hadoop.ParquetFileReader;
import parquet.hadoop.metadata.ParquetMetadata;
import parquet.schema.MessageType;
import parquet.schema.MessageTypeParser;
import parquet.schema.Type;
...
public int run(String[] args) throws Exception {
...

  String writeSchema = "message example {\n" +
  "required int32 x;\n" +
  "required int32 y;\n" +
  "}";
  ExampleOutputFormat.setSchema(
    job,
    MessageTypeParser.parseMessageType(writeSchema));

  job.submit();
```

or it can be extracted from the input file(s) if they are in Parquet format:

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.LocatedFileStatus;
import org.apache.hadoop.fs.RemoteIterator;
...

public int run(String[]
  args) throws Exception {
...

String inputFile = args[0];
  Path parquetFilePath = null;
  // Find a file in case a directory was passed

  RemoteIterator<LocatedFileStatus> it = FileSystem.get(getConf()).listFiles(new
Path(inputFile), true);
  while(it.hasNext()) {
      FileStatus fs = it.next();

    if(fs.isFile()) {
      parquetFilePath = fs.getPath();
      break;
    }
  }
  if(parquetFilePath == null) {
    LOG.error("No file found for " + inputFile);
    return 1;
  }
  ParquetMetadata readFooter =
    ParquetFileReader.readFooter(getConf(), parquetFilePath);
  MessageType schema =
    readFooter.getFileMetaData().getSchema();
  GroupWriteSupport.setSchema(schema, getConf());

  job.submit();
```

Records can then be written in the mapper by composing a Group as value using the Example classes and no key:

```
protected void map(LongWritable key, Text value,
  Mapper<LongWritable, Text, Void, Group>.Context context)
  throws java.io.IOException, InterruptedException {
    int x;
    int y;
    // Extract the desired output values from the input text
    //
    Group group = factory.newGroup()
      .append("x", x)
      .append("y", y);
    context.write(null, group);
  }
}
```

Compression can be set before submitting the job with:

```
ExampleOutputFormat.setCompression(job, codec);
```

The codec should be one of the following:

- `CompressionCodecName.UNCOMPRESSED`
- `CompressionCodecName.SNAPPY`
- `CompressionCodecName.GZIP`

Parquet File Interoperability

Impala has included Parquet support from the beginning, using its own high-performance code written in C++ to read and write the Parquet files. The Parquet JARs for use with Hive, Pig, and MapReduce are available with CDH 4.5 and higher. Using the Java-based Parquet implementation on a CDH release prior to CDH 4.5 is not supported.

A Parquet table created by Hive can typically be accessed by Impala 1.1.1 and higher with no changes, and vice versa. Prior to Impala 1.1.1, when Hive support for Parquet was not available, Impala wrote a dummy SerDes class name into each data file. These older Impala data files require a one-time `ALTER TABLE` statement to update the metadata for the SerDes class name before they can be used with Hive. See the Impala Release Notes for details.

A Parquet file written by Hive, Impala, Pig, or MapReduce can be read by any of the others. Different defaults for file and block sizes, compression and encoding settings, and so on might cause performance differences depending on which component writes or reads the data files. For example, Impala typically sets the HDFS block size to 1 GB and divides the data files into 1 GB chunks, so that each I/O request reads an entire data file.

There may be limitations in a particular release. The following are known limitations in CDH 4:

- The `TIMESTAMP` data type in Parquet files is not supported in Hive, Pig, or MapReduce in CDH 4. Attempting to read a Parquet table created with Impala that includes a `TIMESTAMP` column will fail.
- Parquet has not been tested with HCatalog. Without HCatalog, Pig cannot correctly read dynamically partitioned tables; that is true for all file formats.
- Currently, Impala does not support table columns using nested data types or composite data types such as maps, structs, or arrays. Any Parquet data files that include such types cannot be queried through Impala.

Examples of Java Programs to Read and Write Parquet Files

You can find full examples of Java code at the Cloudera Parquet examples Github repository.

The TestReadParquet.java example demonstrates the "identity" transform. It reads any Parquet data file and writes a new file with exactly the same content.

The TestReadWriteParquet.java example reads a Parquet data file, and produces a new text file in CSV format with the same content.

Maintenance Tasks and Notes

Note: Running Services

When starting, stopping and restarting CDH components, always use the `service (8)` command rather than running scripts in `/etc/init.d` directly. This is important because `service` sets the current working directory to `/` and removes most environment variables (passing only *LANG* and *TERM*) so as to create a predictable environment in which to administer the service. If you run the scripts in `/etc/init.d`, any environment variables you have set remain in force, and could produce unpredictable results. (If you install CDH from packages, `service` will be installed as part of the Linux Standard Base (LSB).)

This section provide instructions and information on the following:

- Starting CDH Services
- Configuring init to Start Core Services
- Stopping CDH Services
- Uninstalling CDH Components
- SSH and HTTPS in a Hadoop Cluster

Starting CDH Services

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

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

START services in this order:

Order	Service	Comments	For instructions and more information
1	ZooKeeper	Cloudera recommends starting ZooKeeper before starting HDFS; this is a **requirement** in a high-availability (HA) deployment. In any case, always start ZooKeeper before HBase.	Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server; Installing ZooKeeper in a Production Environment; HDFS High Availability Initial Deployment; 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; Configuring HDFS High Availability
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

Order	Service	Comments	For instructions and more information
			Availability for the JobTracker (MRv1)
4b	YARN	Start YARN before Hive or Oozie. Do not start YARN if MRv1 is running.	Deploying MapReduce v2 (YARN) on a Cluster
5	HBase		Starting the HBase Master; Deploying HBase in a Distributed Cluster
6	Hive	Start the Hive metastore before starting HiveServer2 and the Hive console.	Installing Hive on page 238
7	Oozie		Starting the Oozie Server
8	Flume 1.x		Running Flume
9	Sqoop		Sqoop Installation and Sqoop 2 Installation on page 345
10	Hue		Hue Installation

Configuring init to Start Core Hadoop System Services

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

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

Important:

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

The `chkconfig` commands to use are:

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

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

Where	Command
On the NameNode	`$ 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 nodes at the same time. This is not recommended; it degrades your performance and may result in an unstable MapReduce cluster deployment.

The `chkconfig` commands to use are:

Where	Command
On the NameNode	`$ 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 node	`$ 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 node	`$ 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 slave	`$ sudo chkconfig hbase-regionserver on`
Hive metastore HiveServer2	Hive server	`$ sudo chkconfig hive-metastore  on` `$ sudo chkconfig hive-server2 on`
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 slave	`$ sudo update-rc.d hbase-regionserver defaults`
Hive metastore HiveServer2	On each Hive client	`$ sudo update-rc.d hive-metastore  defaults` `$ 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`

Stopping Services

Run the following command on every host in the cluster to shut down all Hadoop Common system services that are started by `init` 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, issue the following command on each host::

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

You could also use `ps -fu hdfs` and `ps -fu mapred` to confirm that no processes are running as the `hdfs` or `mapred` user, but additional user names are created by the other components in the ecosystem; checking for the "java" string in the ps output is an easy way to identify any processes that may still be running.

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

STOP system services in this order:

Order	Service	Comments	Instructions
1	Hue		Run the following on the Hue Server machine to stop Hue `sudo service hue stop`
2	Sqoop 1		Run the following on all nodes where it is running:

Order	Service	Comments	Instructions
			`sudo service sqoop-metastore stop`
2	Sqoop 2		Run the following on all nodes where it is running: `$ sudo /sbin/service sqoop2-server stop`
3	Flume 0.9		Stop the Flume Node processes on each node where they are running: `sudo service flume-node stop` Stop the Flume Master `sudo service flume-master stop`
4	Flume 1.x	There is no Flume master	Stop the Flume Node processes on each node where they are running: `sudo service flume-ng-agent stop`
5	Oozie		`sudo service oozie stop`
6	Hive		To stop Hive, exit the Hive console and make sure no Hive scripts are running. Shut down HiveServer2: `sudo service hiveserver2 stop` Shut down the Hive metastore daemon on each client: `sudo service hive-metastore stop` If the metastore is running from the command line, use Ctrl-c to shut it down.
7	HBase	Stop the Thrift server and clients, then shut down the cluster.	To stop the Thrift server and clients: `sudo service hbase-thrift stop` To shut down the cluster, use this command on the master node: `sudo service hbase-master stop` Use the following command on each node hosting a region server:

Order	Service	Comments	Instructions
			`sudo service hadoop-hbase-regionserver stop`

8a	MapReduce v1	Stop Hive and Oozie before stopping MapReduce.	To stop MapReduce, stop the JobTracker service, and stop the Task Tracker on all nodes where it is running. Use the following commands:
			
			`sudo service hadoop-0.20-mapreduce-jobtracker stop`
			
			`sudo service hadoop-0.20-mapreduce-tasktracker stop`

8b	YARN	Stop Hive and Oozie before stopping YARN.	To stop YARN, stop the MapReduce JobHistory service, ResourceManager service, and NodeManager on all nodes where they are running. Use the following commands:
			
			`sudo service hadoop-mapreduce-historyserver stop`
			
			`sudo service hadoop-yarn-resourcemanager stop`
			
			`sudo service hadoop-yarn-nodemanager stop`

9	HttpFS		`sudo service hadoop-httpfs stop`
10	HDFS		To stop HDFS: On the NameNode: `sudo service hadoop-hdfs-namenode stop` On the Secondary NameNode (if used): `sudo service hadoop-hdfs-secondarynamenode stop` On each DataNode: `sudo service hadoop-hdfs-datanode stop`

Order	Service	Comments	Instructions

11	ZooKeeper	Stop HBase and HDFS before stopping ZooKeeper.	To stop the ZooKeeper server, use one of the following commands on each ZooKeeper node:
			`sudo service zookeeper-server stop`
			or
			`sudo service zookeeper stop`

Uninstalling CDH Components

Before uninstalling CDH, stop all Hadoop processes, following the instructions in Stopping Services.

Here are the commands to use to uninstall the Hadoop components on different Linux systems.

Operating System	Commands	Comments
Red-Hat-compatible	`yum remove`	
Debian and Ubuntu	`apt-get remove` or `apt-get purge`	`apt-get` can be run with the `remove` option to remove only the installed packages or with the `purge` option to remove packages and configuration
SLES	`zypper remove`	

Uninstalling from Red Hat, CentOS, and Similar Systems

Component to remove	Command
Mahout	`$ sudo yum remove mahout`
Whirr	`$ sudo yum remove whirr`
Hue	`$ sudo yum remove hue`
Pig	`$ sudo yum remove pig`
Sqoop 1	`$ sudo yum remove sqoop`
Sqoop 2	`$ sudo yum remove sqoop2-server sqoop2-client`
Flume	`$ sudo yum remove flume`
Oozie client	`$ sudo yum remove oozie-client`
Oozie server	`$ sudo yum remove oozie`
Hive	`$ sudo yum remove hive hive-metastore hive-server hive-server2`

Component to remove	Command
HBase	`$ sudo yum remove hadoop-hbase`
ZooKeeper server	`$ sudo yum remove hadoop-zookeeper-server`
ZooKeeper client	`$ sudo yum remove hadoop-zookeeper`
ZooKeeper Failover Controller (ZKFC)	`$ sudo yum remove hadoop-hdfs-zkfc`
HDFS HA Journal Node	`$ sudo yum remove hadoop-hdfs-hadoop-hdfs-journalnode`
Hadoop repository packages	`$ sudo yum remove cloudera-cdh<n>`
HttpFS	`$ sudo yum remove hadoop-httpfs`
Hadoop core packages	`$ sudo yum remove hadoop`

Uninstalling from Debian and Ubuntu

Use the `apt-get` command to uninstall software on Debian and Ubuntu systems. You can use `apt-get remove` or `apt-get purge`; the difference is that `apt-get remove` removes all your configuration data as well as the package files.

Warning:

For this reason, you should `apt-get remove` only with great care, and after making sure you have backed up all your configuration data.

The `apt-get remove` commands to uninstall the Hadoop components from a Debian or Ubuntu system are:

Component to remove	Command
Whirr	`$ sudo apt-get remove whirr`
Hue	`$ sudo apt-get remove hue`
Pig	`$ sudo apt-get remove pig`
Sqoop 1	`$ sudo apt-get remove sqoop`
Sqoop 2	`$ sudo apt-get remove sqoop2-server sqoop2-client`
Flume	`$ sudo apt-get remove flume`
Oozie client	`$ sudo apt-get remove oozie-client`
Oozie server	`$ sudo apt-get remove oozie`
Hive	`$ sudo apt-get remove hive hive-metastore hive-server hive-server2`
HBase	`$ sudo apt-get remove hadoop-hbase`
ZooKeeper server	`$ sudo apt-get remove hadoop-zookeeper-server`
ZooKeeper client	`$ sudo apt-get remove hadoop-zookeeper`

Component to remove	Command
ZooKeeper Failover Controller (ZKFC)	`$ sudo apt-get remove hadoop-hdfs-zkfc`
HDFS HA Journal Node	`$ apt-get remove hadoop-hdfs-hadoop-hdfs-journalnode`
HttpFS	`$ sudo apt-get remove hadoop-httpfs`
Hadoop repository packages	`$ sudo apt-get remove cdh<n>-repository`
Hadoop core packages	`$ sudo apt-get remove hadoop`

Uninstalling from SLES

Component removed	Command
Whirr	`$ sudo zypper remove whirr`
Hue	`$ sudo zypper remove hue`
Pig	`$ sudo zypper remove pig`
Sqoop	`$ sudo zypper remove sqoop`
Sqoop	`$ sudo zypper remove sqoop2-server sqoop2-client`
Flume	`$ sudo zypper remove flume`
Oozie server	`$ sudo zypper remove oozie`
Oozie client	`$ sudo zypper remove oozie-client`
Hive	`$ sudo zypper remove hive hive-metastore hive-server hive-server2`
HBase	`$ sudo zypper remove hadoop-hbase`
ZooKeeper server	`$ sudo zypper remove hadoop-zookeeper-server`
ZooKeeper client	`$ sudo zypper remove hadoop-zookeeper`
ZooKeeper Failover Controller (ZKFC)	`$ sudo zypper remove hadoop-hdfs-zkfc`
HDFS HA Journal Node	`$ sudo zypper remove hadoop-hdfs-hadoop-hdfs-journalnode`
HttpFS	`$ sudo zypper remove hadoop-httpfs`
Hadoop repository packages	`$ sudo zypper remove cloudera-cdh`
Hadoop core packages	`$ sudo zypper remove hadoop`

Additional clean-up

The uninstall commands may not remove all traces of Hadoop from your system. The `apt-get purge` commands available for Debian and Ubuntu systems delete more files than the commands that use the `remove` option but are still not comprehensive. If you want to remove all vestiges of Hadoop from your system, look for the following and remove them manually:

- log files
- modified system configuration files
- Hadoop configuration files in directories under `/etc` such as `hadoop`, `hbase`, `hue`, `hive`, `oozie`, `sqoop`, `zookeeper`, and `zookeeper.dist`
- user/group identifiers
- Oozie and Hue databases
- Documentation packages

SSH and HTTPS in the Hadoop Cluster

SSH and HTTPS can be used to transmit information securely:

- **SSH (Secure Shell)** is a secure shell that usually runs on top of SSL and has a built-in username/password authentication scheme that can be used for secure access to a remote host; it is a more secure alternative to `rlogin` and `telnet`.
- **HTTPS (HTTP Secure)** is HTTP running on top of SSL, adding security to standard HTTP communications.

SSH

It is a good idea to use SSH for remote administration purposes (instead of `rlogin`, for example). But note that it is not used to secure communication among the elements in a Hadoop cluster (DataNode, NameNode, TaskTracker or YARN ResourceManager, JobTracker or YARN NodeManager, or the `/etc/init.d` scripts that start daemons locally).

The Hadoop components use SSH in the following cases:

- The sshfencer component of High Availability Hadoop configurations uses SSH; the `shell` fencing method does not require SSH.
- Whirr uses SSH to enable secure communication with the Whirr cluster in the Cloud. See the Whirr Installation instructions.

HTTPS

Some communication within Hadoop can be configured to use HTTPS. Implementing this requires generating valid certificates and configuring clients to use those certificates. The HTTPS functionality that can be configured in CDH 5 is:

- Encrypted MapReduce Shuffle (both MRv1 and YARN).
- Encrypted Web UIs; the same configuration parameters that enable Encrypted MapReduce Shuffle implement Encrypted Web UIs.

These features are discussed under Configuring Encrypted Shuffle, Encrypted Web UIs, and Encrypted HDFS Transport.

The HDFS Balancer

The HDFS balancer re-balances data across the DataNodes, moving blocks from over-utilized to under-utilized nodes. As the system administrator, you can run the balancer from the command-line as necessary -- for example, after adding new DataNodes to the cluster.

Points to note:

- The balancer requires the capabilities of an HDFS superuser (for example, the `hdfs` user) to run.
- The balancer does not balance between individual volumes on a single DataNode.

- You can run the balancer without parameters, as follows:

  ```
  sudo -u hdfs hdfs balancer
  ```

  > **Note:**
  >
  > If Kerberos is enabled, do not use commands in the form `sudo -u <user> hadoop <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a `keytab`) and then, for each command executed by this user, `$ <command>`

  This runs the balancer with a default threshold of 10%, meaning that the script will ensure that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%. For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that each DataNode's disk usage is between 30% and 50% of that DataNode's disk-storage capacity.
- You can run the script with a different threshold; for example:

  ```
  sudo -u hdfs hdfs balancer -threshold 5
  ```

  This specifies that each DataNode's disk usage must be (or will be adjusted to be) within 5% of the cluster's overall usage.
- You can adjust the network bandwidth used by the balancer, by running the `dfsadmin -setBalancerBandwidth` command before you run the balancer; for example:

  ```
  dfsadmin -setBalancerBandwidth newbandwidth
  ```

  where *newbandwidth* is the maximum amount of network bandwidth, in bytes per second, that each DataNode can use during the balancing operation. For more information about the bandwidth command, see this page.
- 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.

Mountable HDFS

CDH 5 includes a FUSE (Filesystem in Userspace) interface into HDFS. FUSE enables you to write a normal userland application as a bridge for a traditional filesystem interface. The `hadoop-hdfs-fuse` package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. It is assumed that you have a working HDFS cluster and know the hostname and port that your NameNode exposes.

To install fuse-dfs On Red Hat-compatible systems:

```
$ sudo yum install hadoop-hdfs-fuse
```

To install fuse-dfs on Ubuntu systems:

```
$ sudo apt-get install hadoop-hdfs-fuse
```

To install fuse-dfs 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 Configuring Software for HDFS HA 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
```

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.

By default, the CDH 5 package installation creates the `/etc/default/hadoop-fuse` file with a maximum heap size of 128 MB. You can change the JVM minimum and maximum heap size; for example

To change it:

```

export LIBHDFS_OPTS="-Xms64m -Xmx256m"
```

Be careful not to set the minimum to a higher value than the maximum.

For more information, see the help for `hadoop-fuse-dfs`:

```
$ hadoop-fuse-dfs --help
```

Configuring an NFSv3 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.

After mounting HDFS to his or her local filesystem, a user can:

- Browse the HDFS file system through 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.

The subsections that follow provide information on installing and configuring the gateway.

Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper/YaST` or `apt` repository, and install or upgrade CDH 5 and make sure it is functioning correctly. For instructions, see CDH 5 Installation on page 27 and Upgrading to CDH 5 on page 58.

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 Red-Hat-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 Red-Hat-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 395, 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 via 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.

Java Development Kit Installation

Install the Oracle Java Development Kit (JDK) before deploying CDH 5.

- To install the JDK, follow the instructions under Oracle JDK Installation. The completed installation must meet the requirements in the box below.
- If you have already installed a version of the JDK, make sure your installation meets the requirements in the box below.

Note: Requirements

- See CDH 5 Requirements and Supported Versions for the recommended and supported JDK versions.
- If you are deploying CDH on a cluster, the same version of the Oracle JDK must be installed on each node.
- The JDK should be installed in `/usr/java/`*`jdk-version`*.
  - In addition, to make sure the right version of the JDK is always used, follow these directions.

You may be able to install the Oracle JDK with your package manager, depending on your choice of operating system.

See this section for installation instructions.

Oracle JDK Installation

Important:

The Oracle JDK installer is available both as an RPM-based installer for RPM-based systems, and as a binary installer for other systems.

On SLES 11 platforms, do not install or try to use the IBM Java version bundled with the SLES distribution; Hadoop will not run correctly with that version. Install the Oracle JDK by following the instructions below.

To install the Oracle JDK:

1. Download the `.tar.gz` file for one of the supported versions of the Oracle JDK from this page. (This link was correct at the time of writing, but the page is restructured frequently.)
2. Extract the JDK to `/usr/java/`*`jdk-version`*; for example `/usr/java/jdk.1.7.0_`*`nn`*, where is a *nn* is a supported version.
3. In `/etc/default/bigtop-utils`, set `JAVA_HOME` to the directory where the JDK is installed; for example:

```
export JAVA_HOME=/usr/java/default
```

4. Symbolically link the directory where the JDK is installed to `/usr/java/default`; for example:

```
ln -s /usr/java/jdk.1.7.0_nn /usr/java/default
```

Creating a Local Yum Repository

This section explains how to set up a local yum repository which you can then use to install CDH on the machines in your cluster. There are a number of reasons you might want to do this, for example:

- The computers in your cluster may not have Internet access. You can still use yum to do an installation on those machines by creating a local yum repository.
- You may want to keep a stable local repository to ensure that any new installations (or re-installations on existing cluster members) use exactly the same bits.
- Using a local repository may be the most efficient way to distribute the software to the cluster members.

To set up your own internal mirror, do the following.

Note:

Before You Start

These instructions assume you already have the appropriate Cloudera repo file on the system on which you are going to create the local repository. If this is not the case, follow the instructions under To download and install the CDH 5 Package. (Downloading and installing the RPM also downloads the repo file and saves it in `/etc/yum.repos.d`.)

1. On a computer that *does* have Internet access, install a web server such as apache/lighttpd on the machine which will serve the RPMs. The default configuration should work. Make sure the firewall on this web server will let http traffic go through.
2. On the same computer as in the previous step, install the `yum-utils` and `createrepo` packages if they are not already installed (`yum-utils` includes the `reposync` command):

```
sudo yum install yum-utils createrepo
```

3. On the same computer as in the previous steps, download the yum repository into a temporary location. On Red Hat/CentOS 6, you can use a command such as:

```
reposync -r cloudera-cdh5
```

Note:

`cloudera-cdh5` is an example of the name of the repository on your system; the name is usually in square brackets on the first line of the repo file, which in this case is `/etc/yum.repos.d/cloudera-cdh5.repo`.

4. Put all the RPMs into a directory served by your web server. For this example, we'll call it `/var/www/html/cdh/4/RPMS/noarch/` (or `x86_64` or `i386` instead of `noarch`). Make sure you can remotely access the files in the directory you just created (the URL should look like `http://<yourwebserver>/cdh/4/RPMS/`).
5. On your web server, go to `/var/www/html/cdh/4/` and type the following command:

```
createrepo .
```

This will create or update the necessary metadata so yum can understand this new repository (you will see a new directory named `repodata`).

> ▪ **Important:**
>
> Check the permissions of the subdirectories and files under `/var/www/html/cdh/4/`. Make sure they are all readable by your web server user.

6. Edit the repo file you got from Cloudera (see Before You Start) and replace the line starting with `baseurl=` or `mirrorlist=` with `baseurl=http://<yourwebserver>/cdh/4/`
7. Save this modified repo file in `/etc/yum.repos.d/`, and check that you can install CDH through `yum`.

Example:

```
yum update && yum install hadoop
```

Once you have confirmed that your internal mirror works, you can distribute this modified repo file to all your machines, and they should all be able to install CDH without needing access to the Internet. Follow the instructions under CDH 5 Installation.

Using the CDH 5 Maven Repository

If you want to build applications or tools with the CDH 5 components and you are using Maven or Ivy for dependency management, you can pull the CDH 5 artifacts from the Cloudera Maven repository. The repository is available at https://repository.cloudera.com/artifactory/cloudera-repos/. The following is a sample POM (`pom.xml`) file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>

</project>
```

For detailed information for each CDH component, by release, see Using the CDH 5 Maven Repository.

Building RPMs from CDH Source RPMs

This section describes how to build binary packages (RPMs) from published CDH source packages (SRPMs):

- Prerequisites
- Setting up an Environment for Building RPMs
- Building an RPM

Prerequisites

- Oracle Java Development Kit (JDK) version 6.
- Apache Ant version 1.7 or later.
- Apache Maven 3.0 or later.
- The following environment variables must be set: `JAVA_HOME`, `JAVA5_HOME`, `FORREST_HOME`, and `ANT_HOME`.
- Your `PATH` must include the `JAVA_HOME`, `ANT_HOME`, `FORREST_HOME` and maven bin directories.
- If you are using Red Hat or CentOS systems, the `rpmdevtools` package is required for the `rpmdev-setuptree` command used below.

Setting up an environment for building RPMs

Red Hat or CentOS systems

Users of these systems can run the following command to set up their environment:

```
$ rpmdev-setuptree                                    # Creates ~/rpmbuild and
~/.rpmmacros
```

SLES systems

Users of these systems can run the following command to set up their environment:

```
$ mkdir -p ~/rpmbuild/{BUILD,RPMS,S{OURCE,PEC,RPM}S}
$ echo "%_topdir $HOME/rpmbuild">  ~/.rpmmacros
```

Building an RPM

Download SRPMs from archive.cloudera.com. The source RPMs for CDH 5 reside at http://archive.cloudera.com/cdh5/redhat/5/x86_64/cdh/5/SRPMS/, http://archive.cloudera.com/cdh5/sles/11/x86_64/cdh/5/SRPMS/ or http://archive.cloudera.com/cdh5/redhat/6/x86_64/cdh/5/SRPMS/. Run the following commands as a non-root user, substituting the particular SRPM that you intend to build:

```
$ export SRPM=hadoop-0.20-0.20.2+320-1.src.rpm
$ rpmbuild --nodeps --rebuild $SRPM                   # Builds the native RPMs
$ rpmbuild --nodeps --rebuild --target noarch $SRPM  # Builds the java RPMs
```

The built packages can be found in `$HOME/rpmbuild/RPMS`.

Getting Support

This section describes how to get support for CDH 5:

- Cloudera Support
- Community Support
- Report Issues
- Get Announcements about New Releases

Cloudera Support

Cloudera can help you install, configure, optimize, tune, and run Hadoop for large scale data processing and analysis. Cloudera supports Hadoop whether you run our distribution on servers in your own data center, or on hosted infrastructure services such as Amazon EC2, Rackspace, SoftLayer, or VMware's vCloud.

If you are a Cloudera customer, you can:

- Visit the Cloudera Knowledge Base.
- Learn how to register for an account to create a support ticket at the support site.

If you are not a Cloudera customer, learn how Cloudera can help you.

Community Support

Register for the Cloudera Users groups.

If you have any questions or comments about CDH, you can send a message to the CDH user's list: cdh-user@cloudera.org

If you have any questions or comments about using Cloudera Manager, you can send a message to the Cloudera Manager user's list: scm-users@cloudera.org

Report Issues

Cloudera tracks software and documentation bugs and enhancement requests for CDH on issues.cloudera.org. Your input is appreciated, but before filing a request, please search the Cloudera issue tracker and "CDH Manual Installation" Community Forum for existing issues, and send a message to the CDH user's list, cdh-user@cloudera.org, or the CDH developer's list, cdh-dev@cloudera.org.

If you would like to report or view software issues and enhancement requests for Cloudera Manager, visit this site: https://issues.cloudera.org/browse/CM

Get Announcements about New Releases

To get information about releases and updates for all products, sign up for Release Announcements forum.

Apache and Third-Party Licenses

This section describes the licenses that apply to CDH 5:

- Apache
- Third-Party

Apache License

All software developed by Cloudera for CDH is released with an Apache 2.0 license. Please let us know if you find any file that doesn't explicitly state the Apache license at the top and we'll immediately fix it.

Third-Party Licenses

For a list of third-party licenses associated with CDH, see http://www.cloudera.com/content/cloudera-content/cloudera-docs/Licenses/Third-Party-Licenses/Third-Party-Licenses.html.