cloudera

Cloudera, Inc.
210 Portage Avenue
Palo Alto, CA 94306
info@cloudera.com
US: 1-888-789-1488
Intl: 1-650-362-0488
www.cloudera.com

Important Notice

Version: CDH3 Update 2

Date: October 20, 2011

Contents

ABOUT THIS GUIDE 1

WHAT'S NEW IN CDH3 1

What's New in CDH3 Update 2 1

CDH-wide changes for Update 2 1

New Components 1

Updated Components 2

What's New in CDH3 Update 1 3

CDH-wide changes for Update 1 4

Updated Components 4

What's New in CDH3 Update 0 5

CDH-wide changes 6

Updated Components 6

New Components 8

About CDH Package Manifests 9

BEFORE YOU INSTALL CDH3 ON A CLUSTER 9

Supported Operating Systems for CDH3 10

CDH3 INSTALLATION 10

Ways To Install CDH3 10

Before You Begin Installing CDH Manually 11

Installing CDH3 on Red Hat Systems 12

Step 1: Download the CDH3 Repository or Package 12

Step 2: Install CDH3. 13

Installing CDH3 on Ubuntu and other Debian Systems 14

Step 1: Download the CDH3 Repository or Package 14

Step 2: Install CDH3. 16

Installing CDH3 on SUSE Systems 17

Step 1: Download the CDH3 Repository or Package 17

Step 2: Install CDH 18

Installing CDH3 Components 19

Additional Configuration is Required for These CDH Components 21
Viewing the Apache Hadoop Documentation 22
UPGRADING TO CDH3 22
Before You Begin 22
Upgrading Hive and Hue in CDH3 22
Upgrading Flume in CDH3 23
Changes in User Accounts and Groups in CDH3 Due to Security 23
Directory Ownership in Local File System 23
Directory Ownership on HDFS 24
Upgrading to CDH3 Update 2 25
Step 1: Prepare the cluster for the upgrade. 26
Step 2: Upgrade the Hadoop core package hadoop-0.20 on each of the hosts in your cluster. 27
Step 3: Install each type of daemon package on the appropriate host. 31
Step 4: Start the Hadoop services. 32
Step 5: Verify basic cluster operation. 33
Step 6: (Optional) For security, set the sticky bit on directories. 34
Upgrading Components to CDH3 Update 2 34
Upgrading from a CDH release before CDH3 Beta 3 to CDH3 Update 2 35
Step 1: Prepare the cluster for the upgrade. 35
Step 2: Upgrade the Hadoop core package hadoop-0.20 on each of the hosts in your cluster. 37
Step 3: Install each type of daemon package on the appropriate host. 40
Step 4: Verify that the Hadoop environment values are correct. 41
Step 5: Verify the Hadoop system accounts. 42
Step 6: Verify ownership of HDFS data. 42
Step 7: Fix ownership and permissions of the MapReduce local directories. 44
Step 8: Fix ownership and permissions of the log directories. 45
Step 9: Upgrade the HDFS metadata. 45
Step 10: Start HDFS. 48
Step 11: Change ownership of the mapred.system.dir directory on HDFS. 48
Step 12: Start a Secondary NameNode and MapReduce. 49
Step 13: Verify basic cluster operation. 49

Step 14: Finalize the HDFS upgrade. *50*

Step 15: (Optional) For security, set the sticky bit on directories. *51*

CONFIGURING PORTS FOR CDH3 ..... **52**

PORTS USED BY COMPONENTS OF CDH3 52

PORTS USED BY THIRD PARTIES 56

CDH3 DEPLOYMENT IN STANDALONE MODE **57**

CDH3 DEPLOYMENT IN PSEUDO-DISTRIBUTED MODE **59**

CDH3 DEPLOYMENT ON A CLUSTER **62**

CONFIGURATION 63

Customizing the Configuration without Using a Configuration Package *63*

Configuration Files and Properties *66*

Configuring Local Storage Directories for Use by HDFS and MapReduce *67*

Creating and Configuring the mapred.system.dir Directory in HDFS *71*

Deploying your Custom Configuration to your Entire Cluster *71*

INITIAL SETUP 72

Format the NameNode *72*

Configuring Backup for the NameNode *72*

RUNNING SERVICES 73

Starting HDFS *73*

Starting MapReduce *73*

Configuring the Hadoop Daemons to Start at Boot Time *73*

When SSH is and is not Used *74*

FLUME INSTALLATION **74**

UPGRADING FLUME IN CDH3 74

Upgrading Flume to CDH3 Update 2 ..... *75*

FLUME PACKAGING 76

FLUME PREREQUISITES 76

INSTALLING THE FLUME TARBALL (.TGZ) 77

INSTALLING THE FLUME RPM OR DEBIAN PACKAGES 78

STARTING FLUME NODES ON BOOT UP AUTOMATICALLY 79

STARTING THE FLUME MASTER ON BOOT UP AUTOMATICALLY 80

RUNNING FLUME 80

Files Installed by the Flume RPM and Debian Packages 81
Viewing the Flume Documentation 82
SQOOP INSTALLATION 83
Upgrading Sqoop to CDH3 Update 2 83
Sqoop Packaging 84
Sqoop Prerequisites 84
Installing the Sqoop RPM or Debian Packages 84
Installing the Sqoop Tarball 85
Viewing the Sqoop Documentation 86
HUE INSTALLATION 86
Upgrading Hue in CDH3 86
Upgrading Hue from CDH3 Beta 4 to CDH3 Update 1 or Later 87
Upgrading Hue from CDH3 Update 1 to CDH3 Update 2 88
Installing, Configuring, and Starting Hue on One Machine 88
Specifying the Secret Key 89
Starting Hue on One Machine 90
Installing and Configuring Hue on a Cluster 90
Installing Hue on a Cluster 90
Specifying the Secret Key 91
Configuring the Hadoop Plugins for Hue 92
Restarting the Hadoop Daemons 93
Pointing Hue to Your CDH NameNode and JobTracker 93
Web Server Configuration 95
Authentication 95
Listing all Configuration Options 96
Viewing Current Configuration Settings 96
Using Multiple Files to Store Your Configuration 97
Starting and Stopping Hue 97
Hue Process Hierarchy 97
Hue Logging 98
Viewing Recent Log Messages through your Web Browser 98
The Hue Database 99

Inspecting the Hue Database *99*

Backing up the Hue Database *99*

Configuring Hue to Access Another Database *100*

Configuring Hue to Store Data in MySQL *100*

Installing and Configuring Hue Shell 101

Verifying Hue Shell Installation *102*

Modifying the Hue Shell Configuration File *102*

Unix User Accounts *105*

Running the Appropriate Web Server *106*

Viewing the Hue and Hue Shell Documentation 106

PIG INSTALLATION 106

Upgrading Pig to CDH3 Update 2 106

Installing Pig 107

Problem after Upgrading from CDH3 Update 0 to CDH3 Update 2 108

Incompatible Changes as of the Pig 0.7.0 Release 108

Using Pig with HBase 108

Viewing the Pig Documentation 109

OOZIE INSTALLATION 109

Upgrading Oozie to CDH3 Update 2 109

Step 1: Back Up Configuration Files and Database *109*

Step 2: Stop the Oozie Server *110*

Step 3: Install Oozie *110*

Step 4: Move Directories (tarball only) *110*

Step 5: Run the Setup Script *110*

Step 6: Edit the Configuration File *111*

Step 7: Start Oozie *112*

Oozie Packaging 112

Oozie Prerequisites 112

Installing Oozie Tarball 112

Installing Oozie RPM or Debian Packages 113

Configuring Oozie 114

Hadoop Configuration *116*

Database Configuration *116*
Enabling the Oozie Web Console *122*
Configuring Oozie with Kerberos Security *123*
Installing Oozie ShareLib in Hadoop HDFS *123*
STARTING, STOPPING, AND ACCESSING THE OOZIE SERVER 123
Starting the Oozie Server *123*
Stopping the Oozie Server *124*
Accessing the Oozie Server with the Oozie Client *124*
Accessing the Oozie Server with a Browser *125*
VIEWING THE OOZIE DOCUMENTATION 125
HIVE INSTALLATION 125
UPGRADING HIVE IN CDH3 126
Upgrading Hive from CDH3 Update 1 to CDH3 Update 2 ..... *126*
INSTALLING HIVE 127
CONFIGURING THE HIVE METASTORE 127
Prerequisites *127*
Hive Configuration *129*
USING HIVE WITH HBASE 130
VIEWING THE HIVE DOCUMENTATION 130
HBASE INSTALLATION 130
UPGRADING HBASE TO CDH3 UPDATE 2 130
Step 1: Stop the Thrift Server and Clients *131*
Step 2: Stop the Cluster *131*
Step 3: Stop the ZooKeeper Server *131*
Step 4: Install the new version of HBase *132*
INSTALLING HBASE 132
HOST CONFIGURATION SETTINGS FOR HBASE 133
Using DNS with HBase *133*
Using the Network Time Protocol (NTP) with HBase *133*
Setting User Limits for HBase *133*
Using dfs.datanode.max.xcievers with HBase *135*
STARTING HBASE IN STANDALONE MODE 135

Installing the HBase Master for Standalone Operation *135*
Starting the HBase Master *136*
Accessing HBase by using the HBase Shell *137*
USING MAPREDUCE WITH HBASE 137
CONFIGURING HBASE IN PSEUDO-DISTRIBUTED MODE 138
Modifying the HBase Configuration *138*
Creating the /hbase Directory in HDFS *138*
Enabling Servers for Pseudo-distributed Operation *139*
Installing the HBase Thrift Server *140*
DEPLOYING HBASE IN A DISTRIBUTED CLUSTER 141
Choosing where to Deploy the Processes *141*
Configuring for Distributed Operation *141*
TROUBLESHOOTING 142
VIEWING THE HBASE DOCUMENTATION 142
ZOOKEEPER INSTALLATION 143
UPGRADING ZOOKEEPER TO CDH3 UPDATE 2 143
Performing a ZooKeeper Rolling Upgrade *143*
INSTALLING THE ZOOKEEPER PACKAGES 145
Installing the ZooKeeper Base Package *145*
Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server *145*
Installing ZooKeeper in a Production Environment *146*
MAINTAINING A ZOOKEEPER SERVER 147
VIEWING THE ZOOKEEPER DOCUMENTATION 147
WHIRR INSTALLATION 147
UPGRADING WHIRR TO CDH3 UPDATE 2 148
Step 1: Stop the Whirr proxy *148*
Step 2: Update the Properties File *148*
Step 3: Destroy the Cluster *149*
Install the new Version *149*
INSTALLING WHIRR 149
GENERATING AN SSH KEY PAIR 150
DEFINING A WHIRR CLUSTER 150

Launching a Cluster 151
Running a Whirr Proxy 151
Running a MapReduce job 152
Destroying a cluster 153
Viewing the Whirr Documentation 153
SNAPPY INSTALLATION 154
Upgrading Snappy to CDH3 Update 2 154
Snappy Installation 154
Using Snappy for MapReduce Compression 155
Using Snappy for Pig Compression 156
Using Snappy for Hive Compression 156
Configuring Flume to use Snappy Compression 156
Using Snappy compression in Flume Sinks 156
Using Snappy compression in Sqoop Imports 157
Configuring HBase to use Snappy Compression 157
Viewing the Snappy Documentation 158
MAHOUT INSTALLATION 159
Mahout Prerequisites 159
Mahout Installation 159
Mahout Documentation 160
AVRO USAGE 161
Avro Data Files 161
Compression 161
Flume 161
Sqoop 162
MapReduce 162
Streaming 163
Pig 163
Hive 164
Avro Tools 165
MOUNTABLE HDFS 166
JAVA DEVELOPMENT KIT INSTALLATION 168

JDK Installation on Red Hat 5 and 6, CentOS 5 and 6, or SLES 11 Systems 168
JDK Installation on Ubuntu Systems 169
CREATING A LOCAL YUM RESPOSITORY 169
USING THE CDH3 MAVEN REPOSITORY 170
BUILDING RPMS FROM CDH SOURCE RPMS 173
Prerequisites 173
Setting up an environment for building RPMs 174
Red Hat or CentOS systems 174
SUSE systems 174
Building an RPM 174
GETTING SUPPORT 174
Cloudera Support 174
Community Support 175
APACHE AND THIRD-PARTY LICENSES 175
Apache License 175
Third-Party Licenses 175

About this Guide

This *CDH3 Installation Guide* is for Apache Hadoop developers and system administrators interested in Hadoop installation. The following sections describe how to install and configure version 3 of Cloudera's Distribution including Apache Hadoop (CDH3) as a Yum, Apt, or zypper/YaST repository. It also describes how to deploy in standalone mode, pseudo-distributed mode, and on a cluster.

> **Note**
>
> If you want to download and install a tarball, go to Downloads.

What's New in CDH3

CDH3 contains many changes and enhancements when compared to previous releases. For details about the changes in CDH3, see the CDH3 Release Notes. For links to the detailed change lists that describe the bug fixes and improvements to all of the CDH3 projects, see the packaging section of Downloading CDH Releases.

> **Important**
>
> If you are upgrading from a release of CDH prior to CDH3 Beta 3, read the CDH3 Beta 3 release notes as well as the CDH3 Beta 4 release notes. For more information about upgrading, see Upgrading to CDH3.

What's New in CDH3 Update 2

CDH-wide changes for Update 2

- Avro is supported as a file format throughout CDH.
- As of CDH3 Update 2, CDH officially supports CentOS 6. See Supported Operating Systems for CDH3.

New Components

Apache Mahout

- Apache Mahout is newly introduced to CDH (as of CDH3u2) - version 0.5

Updated Components

Apache Hadoop Common, HDFS and MapReduce

- Introduces per-job limits for counter names, number of counters, status messages, and split sizes.
- Introduces a Fair Scheduler option to fail jobs submitted to non-existent pools.
- Adds Kerberos HTTP SPNEGO authentication support to Hadoop JobTracker, NameNode, DataNode, and TaskTracker web consoles.
- Incorporates significant Common, HDFS and MapReduce bug fixes and improvements.

Apache Flume (incubating)

- Fixes several deadlock and race conditions
- Includes support for Hadoop sequence file output file format and Snappy compression
- Upgraded to Avro 1.5.4

Apache Sqoop (incubating)

- Sqoop user guide now contains a troubleshooting section.
- Includes support for `NVARCHAR` datatype.
- Includes mixed mode insert/update support for Oracle.
- Includes support for working with DB2 database.
- Includes support for explicit boundary query for limiting imports.
- Includes support for multiple column names for update during export.
- Incorporates significant bug fixes and improvements.

Hue

- Includes packaging improvements that rework the way Hue is packaged, tighten up dependencies, and resolve other issues.

Apache Pig

- Includes packaging improvements.

Apache Oozie

- Includes support for PostgreSQL 8.4 & 9.0
- Provides new workflow actions: email and distcp

- Includes support for multiple workflow sharelibs
- Incorporates scalability and stability fixes (de-duplication of coordinator action commands)
- Incorporates database purge fixes

Apache Hive

- Incorporates bug fixes and feature enhancements for the Hive JDBC driver.
- Includes packaging improvements.

Apache HBase

- Updated to upstream version 0.90.4
- Includes backports of several new features for monitoring and manageability:
    - Master and Region Server web interfaces now show status of currently executing tasks
    - Web servers now host a servlet with debugging information available at `/dump`
    - `hbck` now supports a `-metaonly` flag to check only metadata tables.
- Fixes several cases in which regions could become stuck in transition, as well as other stability/integrity issues.
- Introduces a new capability that allows one to pre-specify region-splits keys when using the shell to create tables.

All users are encouraged to upgrade.

Apache ZooKeeper

- Improves service availability during mass client disconnect event.
- Includes packaging improvements.

Apache Whirr

- Updated to upstream version 0.5.0
- Support for running CDH Hadoop, HBase, and ZooKeeper

What's New in CDH3 Update 1

This section summarizes the high level changes and most important new features in CDH3 Update 1 as compared to CDH3 Update 0. Refer to the individual project release notes, (found here) for links to the detailed change lists. For upgrade instructions, refer to the Upgrading to CDH3 section in the CDH3 Installation Guide.

If upgrading from a beta release of CDH3, refer to the CDH3 Release History for information on the changes between beta releases.

CDH-wide changes for Update 1

- Snappy-based compression is integrated throughout CDH.
- Adds support for Debian Lenny and Squeeze releases.
- Adds a web shell for Hue, providing access to the Pig, Flume, and HBase command-line shells.
- Flume/HBase integration.
- Fair Scheduler Access Control Lists.
- Hundreds of bug fixes to the code and packaging. See this page for links to lists of bug fixes for all the CDH3 projects.

Updated Components

Apache HDFS and MapReduce

- NameNode stability improvements.
- Better DataNode handling of hard drive failures.
- Stability and performance improvements for Fuse-DFS (Mountable HDFS).

Apache Hive

CDH3u1 updates Apache Hive from version 0.7.0 to 0.7.1. Improvements include:

- Bug fixes and feature enhancements for the Hive JDBC driver.
- Performance enhancements backported from the 0.8.0-dev branch.
- Upgrade and schema creation scripts for PostgreSQL backed Metastores.

Apache Pig

CDH3u1 updates Apache Pig from a base version of 0.8.0 to 0.8.1. Highlighted improvements include:

- Performance and scalability improvements
- Numerous bug fixes

Hue

- CDH3u1 introduces the Hue Shell, a web shell that provides access to the Pig, Flume, and HBase command-line shells.

Sqoop

CDH3u1 updates Sqoop from version 1.2.0 to 1.3.0. Highlighted improvements and new features include:

- Support for importing data in Avro Data File format.
- Support for codec aliases.
- Support for populating Hive partitions.
- Support for dropping Hive delimiters from String data during imports.
- Significant bugfixes and performance improvements.

Apache HBase

CDH3u1 updates Apache HBase from version 0.90.1 to 0.90.3. Highlighted improvements and new features include:

- HBase bulk loading.

Oozie

Highlighted improvements and new features include:

- Added an E-mail action to provide notification of workflow status (success, failure).
- Added date formatting function.

Apache ZooKeeper

Highlighted improvements include:

- Packaging improvements.

Flume

CDH3U1 updates Flume from version 0.9.3 to 0.9.4.

- HBase integration.

What's New in CDH3 Update 0

This section summarizes the high level changes and most important new features in CDH3 Update 0 as compared to CDH2. Refer to the individual project release notes, (found here) for links to the detailed change lists.

If upgrading from a beta release of CDH3, refer to the CDH3 Release History for information on the changes between beta releases.

CDH-wide changes

- CDH3 adds support for the following new host platforms:
    - Red Hat Enterprise Linux (RHEL) 6.0
    - SUSE Linux Enterprise Server (SLES) 11
    - Ubuntu Lucid and Maverick (32- and 64-bit versions)
- CDH3 removes support for the following host platforms:
    - Debian Lenny
    - Ubuntu Hardy, Jaunty, and Karmic.
- Java artifacts for all components are now published to a Cloudera maven repository

Updated Components

Apache HDFS and MapReduce

- HDFS and MapReduce now support Kerberos-based strong authentication and security. A cluster may be configured in a secured mode whereby users must obtain Kerberos credentials before interacting with a cluster. For more details, see the CDH3 Security Guide.
- HDFS and MapReduce include several enhancements relating to authorization and security:
    - The HDFS and MapReduce service daemons now run as separate `hdfs` and `mapred` Unix accounts for improved security isolation.
    - HDFS support for setting the "sticky bit" on directories to allow multiple users to securely share a single directory
    - Improved support for Access Control Lists in MapReduce at the cluster, queue, and job level.
    - The ability to have MapReduce tasks run as the Unix user that submitted the job.
    - Improved capability for audit logging in MapReduce and HDFS.
- Improved HDFS write path to provide better durability for applications like HBase.
- Significantly improved fair scheduler with better behavior under mixed job workloads.
- Significantly improved scheduling throughput for MapReduce jobs with many small tasks.

- Improved scalability and lower memory consumption for the HDFS NameNode and MapReduce JobTracker.
- Removed Sqoop from the core Hadoop project to create a standalone component.
- Removed the "cloud" contrib scripts, superseded in CDH3 by Apache Whirr
- Significant miscellaneous bug fixes and stability improvements included from the upstream Apache Hadoop project.

Apache Hive

CDH3 updates Apache Hive from a base version of 0.4.1 to 0.7.0. New features and improvements include:

- Improved support for SQL features including CREATE TABLE AS SELECT, LEFT SEMI JOIN, and LATERAL VIEW.
- Improved performance for many types of join and aggregation queries.
- Support for views, multiple databases, dynamic partitions, automatic merging of small files, and archiving.
- Support for alternative storage handlers including the ability to load and query data in Apache HBase.
- Support for RCFile, a columnar storage format providing improved performance and storage efficiency.
- Numerous bug fixes and stability improvements.
- Preliminary support for basic authorization and authentication features.

Apache Pig

CDH3 updates Apache Pig from a base version of 0.5.0 to 0.8.0. Highlighted improvements and new features include:

- Performance and memory usage improvements
- A new Accumulator interface for UDFs, as well as the ability to implement UDFs in Python
- New interfaces for defining LoadFuncs and StoreFuncs
- The ability to manipulate scalar types
- The ability to integrate custom partitioners or custom MapReduce steps
- The ability to load and store data in Apache HBase

Hue (formerly Cloudera Desktop)

Cloudera Desktop has been renamed Hue, and CDH3 includes Hue 1.2.0. New features include:

- The ability to run against a secured Apache Hadoop cluster.
- Significant improvements to the Hue SDK
- Improved support for internationalization and extended character sets.
- Many bug fixes

Sqoop

Sqoop has been broken out of the Hadoop component and now exists as a significantly more fully-featured standalone tool. New features include:

- Improved command-line interface and configuration mechanism.
- The ability to export data from HDFS into relational databases as well as Apache HBase.
- Improved performance and reliability when integrating with common relational databases such as MySQL and Oracle.
- Support for large objects (CLOBs and BLOBs)
- The ability to embed Sqoop as part of an Oozie workflow.
- The ability to perform incremental imports of changing datasets.

New Components

This section lists the components of CDH that are newly available in CDH3.

Apache HBase

Apache HBase is a highly scalable distributed datastore capable of managing large amounts of structured or semi-structured data and allowing realtime reads and writes.

CDH3 includes HBase 0.90.1 with some small bug fixes and patches backported from the upcoming HBase 0.90.2 release.

Oozie

Oozie is the Hadoop workflow engine, and allows users to build and deploy complex multi-step data pipelines.

CDH3 includes Oozie 2.3.0.

Apache Whirr

Whirr is a tool for quickly starting and managing clusters running on cloud services like Amazon EC2.

CDH3 includes Apache Whirr 0.3.0.

Apache ZooKeeper

Apache ZooKeeper is a system that provides distributed coordination services for clustered applications. It is used internally by Apache HBase and may also be used to simplify development of distributed services.

CDH3 includes ZooKeeper 3.3.3.

Flume

Flume is a system for reliably and flexibly collecting and transporting data into HDFS or HBase.

CDH3 includes Flume 0.9.3.

About CDH Package Manifests

Both the CDH patched source and packages contain explicit information about Cloudera modifications. For example, in the patched source there is a top-level cloudera directory with:

- A `CHANGES.cloudera.txt` file that lists all the changes to the pristine source
- A `patches` directory that contains every patch Cloudera has applied to the pristine source. All Cloudera patches are released with an Apache 2.0 license.
- A `files` directory for files Cloudera created from scratch, such as man pages and configuration files. All Cloudera files are released with an Apache 2.0 license.
- A `README.cloudera` file explains explicitly how to recreate the patches source from the pristine source.

Before You Install CDH3 on a Cluster

Before you install CDH3 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 CDH3. See the next section, Supported Operating Systems for CDH3.

2. If you haven't already done so, install the 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.

Supported Operating Systems for CDH3

CDH3 supports the following operating systems:

- For Ubuntu systems, Cloudera provides 32-bit and 64-bit packages for Lucid (10.04) and Maverick (10.10).
- For Debian systems, Cloudera provides 32-bit and 64-bit packages for Squeeze (6.0.2) and Lenny (5.0.8).
- For Red Hat systems, Cloudera provides 32-bit and 64-bit packages for Red Hat Enterprise Linux 5 and CentOS 5, and 64-bit packages for Red Hat Enterprise Linux 6 and CentOS 6. Cloudera recommends using update 5 or later of Red Hat Enterprise Linux 5. Cloudera has received reports that our RPMs work well on Fedora, but we have not tested this.
- For SUSE systems, Cloudera provides 64-bit packages for SUSE Linux Enterprise Server 11 (SLES 11). Service pack 1 or later is required.

> **Note**
>
> If you are using an operating system that is not supported by Cloudera's Debian or RPM packages, you can also download source tarballs from Downloads.

CDH3 Installation

Ways To Install CDH3

You can install CDH3 in any of the following ways:

- Automated method using SCM Express; instructions here.
  Service and Configuration Manager Express Edition (SCM Express) automates the installation and configuration of CDH3 on an entire cluster (up to 50 nodes),

requiring only that you have root SSH access to your cluster's machines. For more information, see the SCM Express Documentation.

> **Note**
>
> SCM Express is not supported on the following 32-bit platforms:
>
> - Red Hat
> - Ubuntu or other Debian

- Manual methods described below:
  - Download and install the CDH3 package
  - Add the CDH3 repository
  - Build your own CDH3 repository
- Install from a CDH3 tarball — If you want to download and install a tarball, see Downloads.

The following instructions describe downloading and installing a package, adding a repository, and building your own repository. If you use one of these methods rather than SCM Express, the first (downloading and installing a package) is recommended in most cases because it is simpler than building or adding a repository.

Before You Begin Installing CDH Manually

- This section contains instructions for new installations. If you need to upgrade from an earlier release, see Upgrading to CDH3.
- For a list of supported operating systems, see Supported Operating Systems for CDH3.

> **Important**
>
> If you have not already done so, install the Java Development Kit (JDK). CDH3 requires JDK 1.6, update 8 at a minimum, which you can download from the Java SE Homepage. You may be able to install the JDK with your package manager, depending on your choice of operating system. For JDK installation instructions, see Java Development Kit Installation.

Installing CDH3 on Red Hat Systems

If you are installing CDH3 on a Red Hat system, you can download Cloudera packages using `yum` or your web browser.

Step 1: Download the CDH3 Repository or Package.

Use one of the following methods to download the CDH3 repository or package:

- Download and install the CDH3 Package *or*
- Add the CDH3 U2 repository *or*
- Build a Yum Repository

To download and install the CDH3 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).

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

2. Install the RPM:

```
$ sudo yum --nogpgcheck localinstall cdh3-
repository-1.0-1.noarch.rpm
```

Now continue with Step 2: Install CDH3.

To add the CDH3 repository:

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

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

(To install a different version of CDH on a Red Hat system, open the repo file (for example, cloudera-cdh3.repo, and change the `3` in the repo file to the version number you want. For example, change the `3` to `3u0` to install CDH3 Update 0.)

Now continue with Step 2: Install CDH3.

To build a CDH3 repository:

If you want to create your own `yum` repository, create a mirror of the CDH3 Red Hat directory and then create a `yum` repository from the mirror. Instructions for how to create a Yum repository are here.

Now continue with Step 2: Install CDH3.

Step 2: Install CDH3.

To install CDH3 on a Red Hat system:

Before installing: (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing one of the following commands:

For Red Hat/CentOS 5 systems:

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

For Red Hat/CentOS 6 systems:

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

1. Find and install the Hadoop core package. For example:

```
$ yum search hadoop
$ sudo yum install hadoop-0.20
```

> **Note**
>
> In prior versions of CDH, the `hadoop-0.20` package contained all of the service scripts in `/etc/init.d/`. In CDH3, the `hadoop-0.20` package does not contain any service scripts – instead, those scripts are contained in the `hadoop-0.20-<daemon>` packages. Do the following step to install the daemon packages.

2. Install each type of daemon package on the appropriate machine. For example, install the NameNode package on your NameNode machine:

```
$ sudo yum install hadoop-0.20-<daemon type>
```

where `<daemon type>` is one of the following:

`namenode`
`datanode`
`secondarynamenode`
`jobtracker`
`tasktracker`

3. Install the CDH component(s) that you want to use. See Installing CDH3 Components.

Installing CDH3 on Ubuntu and other Debian Systems

If you are installing CDH3 on a Debian system, you can download the Cloudera packages using `apt` or your web browser.

Step 1: Download the CDH3 Repository or Package.

Use one of the following methods to download the CDH3 repository or package:

- Download and install the CDH3 Package *or*
- Add the CDH3 U1 repository *or*

- Build a Debian Repository

To download and install the CDH3 package:

1. Click one of the following:
   this link for a Squeeze system, *or*
   this link for a Lenny system, *or*
   this link for a Lucid system, *or*
   this link for a Maverick system.
2. Install the package. Do one of the following:
   Choose **Open with** in the download window to use the package manager, *or*
   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 Downloads/cdh3-repository_1.0_all.deb
```

Now continue with Step 2: Install CDH3.

To add the CDH3 repository:

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

```
deb http://archive.cloudera.com/debian <RELEASE>-cdh3
contrib
deb-src http://archive.cloudera.com/debian <RELEASE>-
cdh3 contrib
```

   where:
   <RELEASE> is the name of your distribution, which you can find by running `lsb_release -c`. For example, to install CDH3 for Ubuntu Lucid, use `lucid-cdh3` in the command above.

   (To install a different version of CDH on a Debian system, specify the version number you want in the `<RELEASE>-cdh3` section of the `deb` command. For example, to install CDH3 Update 0 for Ubuntu Maverick, use `maverick-cdh3u0` in the command above.)
2. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

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

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

Now continue with Step 2: Install CDH3.

To build a CDH3 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 2: Install CDH3.

Step 2: Install CDH3.

To install CDH3 on a Debian system:

1. Update the APT package index:

```
$ sudo apt-get update
```

2. Find and install the Hadoop core package by using your favorite APT package manager, such as `apt-get`, `aptitude`, or `dselect`. For example:

```
$ apt-cache search hadoop
$ sudo apt-get install hadoop-0.20
```

> **Note**
>
> In prior versions of CDH, the `hadoop-0.20` package contained all of the service scripts in `/etc/init.d/`. In CDH3, the `hadoop-0.20` package does not contain any service scripts – instead, those scripts are contained in the `hadoop-0.20-<daemon>` packages. Do the following step to install the daemon packages.

3. Install each type of daemon package on the appropriate machine. For example, install the NameNode package on your NameNode machine:

```
$ sudo apt-get install hadoop-0.20-<daemon type>
```

where <daemon type> is one of the following:

namenode
datanode
secondarynamenode
jobtracker
tasktracker

4. Install the CDH component(s) that you want to use. See Installing CDH3 Components.

Installing CDH3 on SUSE Systems

If you are installing CDH3 on a SUSE system, you can download the Cloudera packages using `zypper` or `YaST` or your web browser.

Step 1: Download the CDH3 Repository or Package.

Use one of the following methods to download the CDH3 repository or package:

- Download and install the CDH3 Package *or*
- Add the CDH3 U1 repository *or*
- Build a SUSE Repository

To download and install the CDH3 package:

1. 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 cdh3-repository-1.0-1.noarch.rpm
```

Now continue with Step 2: Install CDH3.

To add the CDH3 repository:

1. Run the following command:

```
$ sudo zypper addrepo -f
http://archive.cloudera.com/sles/11/x86_64/cdh/cloud
era-cdh3.repo
```

   (To install a different version of CDH on a SUSE system, replace the CDH version number in the command above with the one you want. For example, change the `3` to `3u0` to install CDH3 Update 0.)

2. Update your system package index by running:

```
$ sudo zypper refresh
```

Now continue with Step 2: Install CDH3.

To build a CDH3 repository:

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

Now continue with Step 2: Install CDH3.

Step 2: Install CDH

To install CDH3 on a SUSE system:

1. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

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

1. Find and install the Hadoop core package. For example:

```
$ sudo zypper search hadoop
$ sudo zypper install hadoop-0.20
```

2. Install each type of daemon package on the appropriate machine. For example, install the NameNode package on your NameNode machine:

```
$ sudo zypper install hadoop-0.20-<daemon type>
```

where `<daemon type>` is one of the following:

`namenode`
`datanode`
`secondarynamenode`
`jobtracker`
`tasktracker`

3. Install the CDH component(s) that you want to use. See Installing CDH3 Components.

Installing CDH3 Components

CDH3 includes several components that you can install and use with Apache Hadoop:

- Flume — A distributed, reliable, and available service for efficiently moving large amounts of data as the data is produced. This release provides a scalable conduit to shipping data around a cluster and concentrates on reliable logging. The primary use case is as a logging system that gathers a set of log files on every machine in a cluster and aggregates them to a centralized persistent store such as HDFS.
- Sqoop — A tool that imports data from relational databases into Hadoop clusters. Using JDBC to interface with databases, Sqoop imports the contents of tables into a Hadoop Distributed File System (HDFS) and generates Java classes that enable users to interpret the table's schema. Sqoop can also export records from HDFS to a relational database.

- Hue — A graphical user interface to work with CDH. Hue aggregates several applications which are collected into a desktop-like environment and delivered as a Web application that requires no client installation by individual users.
- 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.
- Hive — A powerful data warehousing application built on top of Hadoop which enables you to access your data using Hive QL, a language that is similar to SQL.
- 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.
- Zookeeper — A highly reliable and available service that provides coordination between distributed processes.
- Oozie — A server-based workflow engine specialized in running workflow jobs with actions that execute Hadoop jobs. A command line client is also available that allows remote administration and management of workflows within the Oozie server.
- Whirr — Provides a fast way to run cloud services.
- Snappy — A compression/decompression library. You do not need to install Snappy if you are already using the native library, but you do need to configure it; see Snappy Installation for more information.
- Mahout — 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.

To install the CDH3 components:

On Red Hat systems, type:

```
$ sudo yum install <CDH3-component-name>
```

On SUSE systems, type:

```
$ sudo zypper install <CDH3-component-name>
```

On Debian systems, type:

```
$ sudo apt-get install <CDH3-component-name>
```

where:
`<CDH3-component-name>` is one of the component names in this table:

To install this CDH3 component	Use this CDH3 component name
Flume	`flume`
Sqoop	`sqoop`
Hue	`hue`
Pig	`hadoop-pig`
Hive	`hadoop-hive`
HBase	`hadoop-hbase`
ZooKeeper	`hadoop-zookeeper`
Oozie server	`oozie`
Oozie client	`oozie-client`
Whirr	`whirr`
Snappy	`hadoop-0.20-native`
Mahout	`mahout`

Additional Configuration is Required for These CDH Components

The following CDH components require additional configuration after installation.

- Hue. For more information, see "Hue Installation" in this guide.
- HBase. For more information, see "HBase Installation" in this guide.
- ZooKeeper. For more information, see "ZooKeeper Installation" in this guide.

- Oozie. For more information, see "Oozie Installation" in this guide.

Viewing the Apache Hadoop Documentation

For additional Apache Hadoop documentation, see http://archive.cloudera.com/cdh/3/hadoop/.

Upgrading to CDH3

The following instructions describe how to upgrade to the Update 2 release of CDH3 (CDH3u2) from an earlier CDH3 release. Cloudera encourages you to upgrade to CDH3 Update 2 to take advantage of bug fixes and extensive testing.

Before You Begin

- If you are running a pseudo-distributed (single-machine) Apache Hadoop cluster, Cloudera recommends that you copy your data off the cluster, remove the old CDH release, install Hadoop 0.20 from CDH3, and then restore your data. Note the information in Upgrading Hive and Hue in CDH3.
- If you have a multi-machine cluster with important data on it, read the following sections to learn how to upgrade your cluster to CDH3.

> **Note**
>
> 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 Hive and Hue in CDH3

If you used Hive or the embedded Hive MetaStore functionality of Beeswax in Hue in versions prior to CDH3 Beta 4, you should be aware that CDH3 includes changes in the Hive MetaStore schema that are part of the Hive 0.7 release. If you want to use Hive or Beeswax in Hue in CDH3, it is imperative that you upgrade the Hive MetaStore schema by running the appropriate schema upgrade script. For information about using the scripts for Hive, see Hive Installation. For information about using the scripts for Hue, see Hue Installation.

If you are upgrading to CDH3 Update 2, make sure you also upgrade Hue, following instructions under Upgrading Hue in CDH3. See also the Hue section under Incompatible Changes.

Upgrading Flume in CDH3

Flume version 0.9.4 in CDH3u1 and later includes two new properties in the `flume-conf.xml` file. To prevent failures from occurring in Flume's internal web applications after an upgrade from CDH3u0 or earlier, you must either add the new properties to your existing Flume 0.9.3 `flume-conf.xml` configuration file or overwrite it with the new Flume 0.9.4 `flume-conf.xml` file. See Flume Installation for more information.

Changes in User Accounts and Groups in CDH3 Due to Security

During CDH3 package installation, two new Unix user accounts called `hdfs` and `mapred` are automatically created to support security:

This User	Runs These Hadoop Programs
`hdfs`	NameNode, DataNodes, and Secondary NameNode
`mapred`	JobTracker and TaskTrackers

The `hdfs` user also acts as the HDFS superuser.

If you upgrade your cluster from CDH3 Beta 2 or earlier to CDH3, during package installation the `hadoop` user is automatically renamed as `hdfs` and the `mapred` user is created. The `hadoop` user no longer exists in CDH3. If you currently use the `hadoop` user to run applications as an HDFS super-user, you should instead use the new `hdfs` user, or instead create a separate Unix account for your application such as `myhadoopapp`.

Directory Ownership in Local File System

Because the HDFS and MapReduce services now run as different users, you must be sure to configure the correct directory ownership of the following files on the local file system of each host:

Directory	Owner	Permissions (see Footnote [1])
`dfs.name.dir`	`hdfs:hadoop`	drwx------
`dfs.data.dir`	`hdfs:hadoop`	drwx------
`mapred.local.dir`	`mapred:hadoop`	drwxr-xr-x

Footnotes:
[1] In CDH3, the Hadoop daemons will automatically configure the correct permissions for you during startup if you configure the directory ownership correctly as shown in the table above.

You must also configure the following permissions for the log directory (`HADOOP_LOG_DIR` in `hadoop-env.sh`) (the default location is `/var/log/hadoop-0.20`), and the `userlogs/` directory:

Directory	Owner	Permissions
`HADOOP_LOG_DIR`	`anyuser:hadoop` [1]	drwxrwxr-x
`userlogs` directory in `HADOOP_LOG_DIR`	`mapred:anygroup`	*permissions will be set automatically at daemon start time*

Footnote:
[1] The `HADOOP_LOG_DIR` directory must have write permissions for the `hadoop` group.

Directory Ownership on HDFS

The following directories on HDFS must also be configured as follows:

Directory	Owner	Permissions
`mapred.system.dir`	`mapred:hadoop`	drwx------ [1]
/ (root directory)	`hdfs:hadoop`	drwxr-xr-x

Footnote:
[1] When starting up, MapReduce sets the permissions for the `mapred.system.dir` directory in HDFS, assuming the user `mapred` owns that directory.

Changing the Directory Ownership on HDFS

To change the directory ownership on HDFS, run the following commands. Replace the example `/mapred/system` directory in the commands below with the HDFS directory specified by the `mapred.system.dir` property in the `conf/mapred-site.xml` file:

- If Hadoop security is not enabled:

```
$ sudo -u hdfs hadoop fs -chown mapred:hadoop
/mapred/system
$ sudo -u hdfs hadoop fs -chown hdfs:hadoop /
$ sudo -u hdfs hadoop fs -chmod -R 700
/mapred/system
$ sudo -u hdfs hadoop fs -chmod 755 /
```

- If Hadoop security is enabled, use `kinit hdfs` to obtain Kerberos credentials for the `hdfs` user when running the following commands:

```
$ kinit -k -t hdfs.keytab
hdfs/fully.qualified.domain.name@YOUR-REALM.COM
$ hadoop fs -chown mapred:hadoop /mapred/system
$ hadoop fs -chown hdfs:hadoop /
$ hadoop fs -chmod -R 700 /mapred/system
$ hadoop fs -chmod 755 /
```

Upgrading to CDH3 Update 2

The procedure you must use to upgrade to CDH3 Update 2 depends on your current CDH version. Upgrading from CDH3 Beta 3, Beta 4, Update 0, or Update 1 is a shorter procedure than upgrading from a version prior to CDH3 Beta 3. This is because the steps to accommodate the Beta 3 security changes were already done in the previous upgrade to CDH3 Beta 3, Beta 4, Update 0 or Update 1.

Follow the instructions in the appropriate section below for your current version:

- **Is CDH3 Beta 3, Beta 4, Update 0 or Update 1 currently installed?** — Follow the steps in this section.
- **Is CDH2, CDH3 Beta 1, or CDH3 Beta 2 currently installed?** — Follow the steps under Upgrading from a CDH release before CDH3 Beta 3 to CDH3 Update 2.

If CDH3 Beta 3, Beta 4, Update 0 or Update 1 is currently installed, use the instructions that follow to upgrade to CDH3 Update 2.

Step 1: Prepare the cluster for the upgrade.

1. Shutdown Hadoop services across your entire cluster by running the following command on every host in your cluster:

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

2. Check each host to make sure that there are no processes running as the `hdfs` or `mapred` users from root:

```
# su hdfs -s /bin/bash -c "jps"
# su mapred -s /bin/bash -c "jps"
```

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

3. Back up the HDFS metadata on the NameNode machine.

> Cloudera recommends backing up HDFS metadata on a regular basis, as well as before a major upgrade.

4. Find the location of your `dfs.name.dir`:

```
$ grep -C1 dfs.name.dir /etc/hadoop/conf/hdfs-
```

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

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 your output is missing any of the files listed above, your backup is not complete. Additionally, 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: Upgrade the Hadoop core package hadoop-0.20 on each of the hosts in your cluster.

For this step, follow the instructions for your operating system:

Instructions for Red Hat systems
Instructions for Ubuntu systems
Instructions for SUSE systems

On Red Hat systems:

1. Delete the `cloudera-cdh3.repo` file in `/etc/yum.repos.d/`.
2. Download the CDH3 Package:
   a. 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).

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

   b. Install the RPM:

```
$ sudo yum --nogpgcheck localinstall cdh3-
repository-1.0-1.noarch.rpm
```

3.

> **Note**
>
> For instructions on how to add a CDH3 yum repository or build your own CDH3 yum repository, see Installing CDH3 on Red Hat Systems.

3. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

   For Red Hat/CentOS 5 systems:

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

For Red Hat 6 systems:

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

4. Find and install the Hadoop core package.

```
$ yum search hadoop
$ sudo yum install hadoop-0.20
```

On Ubuntu systems:

1. Download the CDH3 Package:
   a. Click one of the following:
   this link for a Squeeze system, *or*
   this link for a Lenny system, *or*
   this link for a Lucid system, *or*
   this link for a Maverick system.
   b. Install the package. Do one of the following:
   Choose **Open with** in the download window to use the package manager, *or*
   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 Downloads/cdh3-
repository_1.0_all.deb
```

2.

> **Note**
>
> For instructions on how to add a CDH3 repository or build your own CDH3 repository, see Installing CDH3 on Ubuntu Systems.

2. Update the APT package index:

```
$ sudo apt-get update
```

3. Find and install the Hadoop core package:

```
$ apt-cache search hadoop
$ sudo apt-get install hadoop-0.20
```

On SUSE systems:

1. Delete the `cloudera-cdh3.repo` file.
2. Download the CDH3 Package:
    a. Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).
    b. Install the RPM:

```
$ sudo rpm -i cdh3-repository-1.0-1.noarch.rpm
```

3.

> **Note**
>
> For instructions on how to add a CDH3 repository or build your own CDH3 repository, see Installing CDH3 on SUSE Systems.

3. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

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

4. Update your system package index by running:

```
$ sudo zypper refresh
```

5. Find and install the Hadoop core package. For example:

```
$ sudo zypper search hadoop
$ sudo zypper install hadoop-0.20
```

Step 3: Install each type of daemon package on the appropriate host.

> **Note**
>
> In prior versions of CDH, the `hadoop-0.20` package contained all of the service scripts in `/etc/init.d/`. In CDH3, the `hadoop-0.20` package does not contain any service scripts – instead, those scripts are contained in the `hadoop-0.20-<daemon>` packages. Do this step to install the daemon packages.

Install each type of daemon package on the appropriate host. For example, install the NameNode package on the NameNode machine.

For this step, follow the instructions for your operating system:

Instructions for Red Hat systems
Instructions for Ubuntu systems
Instructions for SUSE systems

On Red Hat systems:

```
$ sudo yum install hadoop-0.20-<daemon type>
```

On Ubuntu systems:

```
$ sudo apt-get install hadoop-0.20-<daemon type>
```

On SUSE systems:

```
$ sudo zypper install hadoop-0.20-<daemon type>
```

where `<daemon type>` is one of the following:

`namenode`
`datanode`
`secondarynamenode`
`jobtracker`
`tasktracker`

Your NameNode is now in safe mode waiting for the DataNodes to connect.

Step 4: Start the Hadoop services.

1. Start the NameNode.

```
$ sudo /etc/init.d/hadoop-0.20-namenode start
```

2. Start the rest of HDFS by running the following command on all the slave nodes:

```
$ sudo /etc/init.d/hadoop-0.20-datanode start
```

3. When the DataNodes start, they will connect to the NameNode. After your DataNodes have started, the NameNode should automatically exit safe mode and HDFS will be in full operation.
4. At this point, you can verify the operation of your HDFS instance by browsing your files, uploading new files, and so on.
5. This is a good time to start a Secondary NameNode using its associated `init.d` script:

```
$ sudo /etc/init.d/hadoop-0.20-secondarynamenode
start
```

6. After you have verified HDFS is operating correctly, you are ready to start MapReduce.

   On the master host, run this command:

```
$ sudo /etc/init.d/hadoop-0.20-jobtracker start
```

   On the slaves, run this command:

```
$ sudo /etc/init.d/hadoop-0.20-tasktracker start
```

7. Verify that the Job Tracker and Task Tracker started properly.

```
ps -eaf | grep -i job
ps -eaf | grep -i task
```

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

Step 5: Verify basic cluster operation.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running a simple pi calculation job.

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
```

2. Run the pi calculation job as the user Joe:

```
$ hadoop-0.20 jar /usr/lib/hadoop-0.20/hadoop-
0.20.2*examples.jar pi 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 38.572 seconds
Estimated value of Pi is 3.14120000000000000000
```

You have now confirmed your cluster is successfully running on CDH3.

> **Important**
>
> If you have client hosts, make sure you also update them to CDH3, including the other CDH packages in the CDH distribution: Apache Pig, Apache Hive, Apache HBase, Flume, Sqoop, Hue, Oozie, Apache ZooKeeper, and Apache Whirr.

Step 6: (Optional) For security, set the sticky bit on directories.

This step is optional and strongly recommended for security. In CDH3, HDFS file permissions have support for the sticky bit. The sticky bit can be set on directories, preventing anyone except the superuser, directory owner, or file owner from deleting or moving the files within the directory. Setting the sticky bit for a file has no effect. This is useful for directories such as `/tmp` which previously had to be set to be world-writable.

To set the sticky bit on the `/tmp` directory, run the following command:

```
$ sudo -u hdfs hadoop fs -chmod 1777 /tmp
```

After running this command, the permissions on `/tmp` will appear as shown below. (Note the "`t`" instead of the final "`x`".)

```
$ hadoop fs -ls /
Found 2 items
drwxrwxrwt - hdfs supergroup 0 2011-02-14 15:55 /tmp
drwxr-xr-x - hdfs supergroup 0 2011-02-14 14:01 /user
```

Upgrading Components to CDH3 Update 2

To upgrade CDH components, see the following sections:

- Flume. For more information, see "Upgrading Flume in CDH3" under "Flume Installation" in this guide.
- Sqoop. For more information, see "Upgrading Sqoop to CDH3 Update 2" under "Sqoop Installation" in this guide.

- Hue. For more information, see "Upgrading Hue in CDH3" under "Hue Installation" in this guide.
- Pig. For more information, see "Upgrading Pig to CDH3 Update 2" under "Pig Installation" in this guide.
- HBase. For more information, see "Upgrading HBase to CDH3 Update 2" under "HBase Installation" in this guide.
- ZooKeeper. For more information, see "Upgrading ZooKeeper to CDH3 Update 2" under "ZooKeeper Installation" in this guide.
- Oozie. For more information, see "Upgrading Oozie to CDH3 Update 2" under "Oozie Installation" in this guide.
- Whirr. For more information, see "Upgrading Whirr to CDH3 Update 2" under "Whirr Installation" in this guide.
- Snappy. For more information, see "Upgrading Snappy to CDH3 Update 2" under "Snappy Installation" in this guide.

Components that are not in the above list have not changed (or are new) for CDH3 U2.

Upgrading from a CDH release before CDH3 Beta 3 to CDH3 Update 2

If CDH2, CDH3 Beta 1, or CDH3 Beta 2 is currently installed, use the instructions in this section to upgrade to CDH3 Update 1.

Important

Before performing the following upgrade procedure, make sure you log in using a user account *other than* the `hadoop` user. Because the `hadoop` user is automatically renamed as `hdfs` during package installation and no longer exists in CDH3, the following upgrade process will fail if you are logged in as `hadoop`.

Step 1: Prepare the cluster for the upgrade.

1. Shutdown Hadoop services across your entire cluster by running the following command on every host in your cluster:

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

```
done
```

2. Check each host to make sure that there are no processes running as user *hadoop*:

```
$ sudo su hadoop -s /bin/bash -c "jps"
```

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

3. Back up the HDFS metadata on the NameNode machine.

> Cloudera recommends backing up HDFS metadata on a regular basis, as well as before a major upgrade.

4. Find the location of your `dfs.name.dir`:

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

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 your output is missing any of the files listed above, your backup is not complete. Additionally, 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: Upgrade the Hadoop core package hadoop-0.20 on each of the hosts in your cluster.

For this step, follow the instructions for your operating system:

Instructions for Red Hat systems
Instructions for Ubuntu systems
Instructions for SUSE systems

On Red Hat systems:

1. Delete the `cloudera-cdh3.repo` file in `/etc/yum.repos.d/`.
2. Download the CDH3 Package:
    a. 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).

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

b. Install the RPM:

```
$ sudo yum --nogpgcheck localinstall cdh3-
repository-1.0-1.noarch.rpm
```

3.

> **Note**
>
> For instructions on how to add a CDH3 yum repository or build your own CDH3 yum repository, see Installing CDH3 on Red Hat Systems.

3. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

   For Red Hat/CentOS 5 systems:

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

   For Red Hat/CentOS 6 systems:

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

4. Find and install the Hadoop core package.

```
$ yum search hadoop
$ sudo yum install hadoop-0.20
```

On Ubuntu systems:

1. Download the CDH3 Package:

a. Click one of the following:
this link for a Squeeze system, *or*
this link for a Lenny system, *or*
this link for a Lucid system, *or*
this link for a Maverick system.

b. Install the package. Do one of the following:
Choose **Open with** in the download window to use the package manager, *or*
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 Downloads/cdh3-
repository_1.0_all.deb
```

2.

> **Note**
>
> For instructions on how to add a CDH3 repository or build your own CDH3 repository, see Installing CDH3 on Ubuntu Systems.

2. Update the APT package index:

```
$ sudo apt-get update
```

3. Find and install the Hadoop core package:

```
$ apt-cache search hadoop
$ sudo apt-get install hadoop-0.20
```

On SUSE systems:

1. Delete the `cloudera-cdh3.repo` file.
2. Download the CDH3 Package:
    a. Click this link, choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).

b. Install the RPM:

```
$ sudo rpm -i cdh3-repository-1.0-1.noarch.rpm
```

3.

Note

For instructions on how to add a CDH3 repository or build your own CDH3 repository, see Installing CDH3 on SUSE Systems.

3. (Optionally) add a repository key. Add the Cloudera Public GPG Key to your repository by executing the following command:

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

4. Update your system package index by running:

```
$ sudo zypper refresh
```

5. Find and install the Hadoop core package. For example:

```
$ sudo zypper search hadoop
$ sudo zypper install hadoop-0.20
```

Step 3: Install each type of daemon package on the appropriate host.

Note

In prior versions of CDH, the `hadoop-0.20` package contained all of the service scripts in `/etc/init.d/`. In CDH3, the `hadoop-0.20` package does not contain any service scripts – instead, those scripts are contained in the `hadoop-0.20-<daemon>` packages. Do the following step to install the daemon packages.

Install each type of daemon package on the appropriate host. For example, install the NameNode package on the NameNode machine:

Instructions for Red Hat systems
Instructions for Ubuntu systems
Instructions for SUSE systems

On Red Hat systems:

```
$ sudo yum install hadoop-0.20-<daemon type>
```

On Ubuntu systems:

```
$ sudo apt-get install hadoop-0.20-<daemon type>
```

On SUSE systems:

```
$ sudo zypper install hadoop-0.20-<daemon type>
```

where `<daemon type>` is one of the following:

`namenode`
`datanode`
`secondarynamenode`
`jobtracker`
`tasktracker`

Step 4: Verify that the Hadoop environment values are correct.

Verify that the values in `hadoop-env.sh` are correct. For example, if a setting is specified for `HADOOP_MASTER` in `hadoop-env.sh`, make sure that setting is correct and the correct source files exist in the `HADOOP_MASTER` location.

Step 5: Verify the Hadoop system accounts.

In CDH3, Hadoop no longer runs as a single `hadoop` user. In order to provide better isolation and security, the HDFS daemons now run as the `hdfs` user and the MapReduce daemons now run as the `mapred` user. Package installation renames the `hadoop` user to `hdfs` and creates a new `mapred` user. Additionally, both users are part of the common `hadoop` group. You can verify that this has occurred correctly using the `id` command:

```
$ id hdfs
uid=116(hdfs) gid=136(hdfs) groups=136(hdfs),123(hadoop)
$ id mapred
uid=125(mapred) gid=137(mapred)
groups=137(mapred),123(hadoop)
```

Note that the numeric user IDs (116 and 125 above) may differ on your system.

> **Note**
>
> Because there is no longer a `hadoop` user in CDH3, you must re-apply any system configuration that was previously applied to that user. For example, if you configured a special file descriptor limit in `/etc/security/limits.conf`, you must update that configuration to apply to both the `hdfs` and `mapred` users.

Step 6: Verify ownership of HDFS data.

Once you have verified that the user accounts have been created by the CDH packaging, you must be sure that your HDFS and MapReduce data directories are owned by the correct users. Your data directories will differ depending on how you have configured Hadoop on your cluster. In this example, the cluster uses the following configurations in `hdfs-site.xml`:

```
 <property>
    <name>dfs.name.dir</name>
    <value>/data/1/dfs/nn,/data/2/dfs/nn</value>
</property>
<property>
    <name>dfs.data.dir</name>

<value>/data/1/dfs/dn,/data/2/dfs/dn,/data/3/dfs/dn,/data
```

```
/4/dfs/dn</value>
</property>
```

Note

If you specified non-existing folders for the `dfs.data.dir` property in the `conf/hdfs-site.xml` file, CDH3 will shut down. (In previous releases, CDH3 silently ignored non-existing folders for `dfs.data.dir`.)

The directories should already be owned by the `hdfs` user:

```
$ ls -ld /data/{1,2,3,4}/dfs/dn /data/{1,2}/dfs/nn
drwxrwxr-x 7 hdfs hadoop 4096 Oct 10 11:25 /data/1/dfs/dn
drwxrwxr-x 7 hdfs hadoop 4096 Oct 10 11:25 /data/2/dfs/dn
drwxrwxr-x 7 hdfs hadoop 4096 Oct 10 11:25 /data/3/dfs/dn
drwxrwxr-x 7 hdfs hadoop 4096 Oct 10 11:25 /data/4/dfs/dn
drwxrwxr-x 6 hdfs hadoop 4096 Oct 10 11:43 /data/1/dfs/nn
drwxrwxr-x 6 hdfs hadoop 4096 Oct 10 11:43 /data/2/dfs/nn
```

If this is not true, use `chown -R` to correct the ownership now.

```
$ sudo chown -R hdfs:hadoop /data/{1,2,3,4}/dfs/dn
```

Note

This process should be done on every host in the cluster.

Important

In versions of Hadoop before CDH3 Beta 3, the default path for the `dfs.data.dir` was `/var/lib/hadoop-0.20/cache/hadoop/dfs/data`. In CDH3 Beta 3 and later, the default path changed to `/var/lib/hadoop-0.20/cache/hdfs/dfs/data`. Similarly, the default path for the `dfs.name.dir` also changed from

`/var/lib/hadoop-0.20/cache/hadoop/dfs/name` to `/var/lib/hadoop-0.20/cache/hdfs/dfs/name` in CDH3 Beta 3 and later.

If your cluster is configured to use the default paths for `dfs.data.dir` or `dfs.name.dir`, you must either move your directories to the new default path locations, or explicitly set the `dfs.data.dir` or `dfs.name.dir` properties in `hdfs-site.xml` to the old default paths.

Step 7: Fix ownership and permissions of the MapReduce local directories.

You must check and fix the permissions of the MapReduce local directories, known as `mapred.local.dir` (See Changes in User Accounts and Groups in CDH3 Due to Security.). This example cluster uses the following configuration:

```
 <property>
    <name>mapred.local.dir</name>

<value>/data/1/mapred/local,/data/2/mapred/local,/data/3/
mapred/local,/data/4/mapred/local</value>
 </property>
```

After the package upgrade, the directories may look like this:

```
$ ls -ld /data/{1,2,3,4}/mapred/local/
drwxrwxr-x 3 hdfs hadoop 4096 Oct  8 16:54
/data/1/mapred/local/
drwxrwxr-x 3 hdfs hadoop 4096 Oct  8 16:54
/data/2/mapred/local/
drwxrwxr-x 3 hdfs hadoop 4096 Oct  8 16:54
/data/3/mapred/local/
drwxrwxr-x 3 hdfs hadoop 4096 Oct  8 16:54
/data/4/mapred/local/
```

In order to fix the ownership, `chown` the directories to the `mapred` user:

```
$ sudo chown -R mapred /data/{1,2,3,4}/mapred
```

Step 8: Fix ownership and permissions of the log directories.

1. Fix ownership and permissions of the `hadoop.log.dir` directory. (See Changes in User Accounts and Groups in CDH3 Due to Security.) This example cluster assumes the default value `/var/log/hadoop-0.20` as `hadoop.log.dir` directory.

   In order to fix the ownership, `chgrp` the `hadoop.log.dir` directory to the `hadoop` group and make sure it is group writable:

   ```
   $ sudo chgrp hadoop /var/log/hadoop-0.20
   $ sudo chmod g+w /var/log/hadoop-0.20
   ```

2. Fix ownership and permissions of the `userlogs` directory in `hadoop.log.dir` to `mapred:anygroup`. (See Changes in User Accounts and Groups in CDH3 Due to Security.)
   This example cluster assumes the default value `/var/log/hadoop-0.20/userlogs` as `userlogs` directory.

   In order to fix the ownership, `chown` the `userlogs` directory to `mapred`:

   ```
   $ sudo chown mapred /var/log/hadoop-0.20/userlogs
   ```

Step 9: Upgrade the HDFS metadata.

1. Start a tail of the NameNode upgrade process by running this command in a separate terminal window:

   ```
   $ tail -f /var/log/hadoop-0.20/hadoop-hadoop-
   namenode*
   ```

2. Upgrade the HDFS metadata.
   Because CDH3 changes the metadata format since earlier versions of CDH, you will need to upgrade your NameNode. To do this, run the following command:

```
$ sudo /etc/init.d/hadoop-0.20-namenode upgrade
```

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

3. In the terminal window running the tail, you should see messages similar to the following:

```
/************************************************
*********
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host =
localhost.localdomain/127.0.0.1
STARTUP_MSG:   args = [-upgrade]
STARTUP_MSG:   version = 0.20.2+737
STARTUP_MSG:   build =  -r
98c55c28258aa6f42250569bd7fa431ac657bdbd; compiled
by 'root' on Mon Oct 11 13:14:05 EDT 2010
************************************************
********/
2010-12-14 00:32:38,776 INFO
org.apache.hadoop.metrics.jvm.JvmMetrics:
Initializing JVM Metrics with processName=NameNode,
sessionId=null
2010-12-14 00:32:38,804 INFO
org.apache.hadoop.hdfs.server.namenode.metrics.NameN
odeMetrics: Initializing NameNodeMeterics using
context
object:org.apache.hadoop.metrics.spi.NoEmitMetricsCo
ntext
2010-12-14 00:32:39,004 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
fsOwner=hdfs
2010-12-14 00:32:39,004 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
supergroup=supergroup
2010-12-14 00:32:39,005 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
isPermissionEnabled=false
```

```
2010-12-14 00:32:39,017 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
isAccessTokenEnabled=false accessKeyUpdateInterval=0
min(s), accessTokenLifetime=0 min(s)
2010-12-14 00:32:39,285 INFO
org.apache.hadoop.hdfs.server.namenode.metrics.FSNam
esystemMetrics: Initializing FSNamesystemMetrics
using context
object:org.apache.hadoop.metrics.spi.NoEmitMetricsCo
ntext
2010-12-14 00:32:39,286 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
Registered FSNamesystemStatusMBean
2010-12-14 00:32:39,315 INFO
org.apache.hadoop.hdfs.server.common.Storage: Number
of files = 24
2010-12-14 00:32:39,325 INFO
org.apache.hadoop.hdfs.server.common.Storage: Number
of files under construction = 0
2010-12-14 00:32:39,325 INFO
org.apache.hadoop.hdfs.server.common.Storage: Image
file of size 2475 loaded in 0 seconds.
2010-12-14 00:32:39,325 INFO
org.apache.hadoop.hdfs.server.common.Storage: Edits
file /var/lib/hadoop-
0.20/cache/hadoop/dfs/name/current/edits of size 4
edits # 0 loaded in 0 seconds.
2010-12-14 00:32:39,326 INFO
org.apache.hadoop.hdfs.server.common.Storage:
Upgrading image directory /var/lib/hadoop-
0.20/cache/hadoop/dfs/name.
   old LV = -19; old CTime = 0.
   new LV = -19; new CTime = 1292315559326
2010-12-14 00:32:39,389 INFO
org.apache.hadoop.hdfs.server.common.Storage: Image
file of size 2475 saved in 0 seconds.
2010-12-14 00:32:39,538 INFO
org.apache.hadoop.hdfs.server.common.Storage:
Upgrade of /var/lib/hadoop-
0.20/cache/hadoop/dfs/name is complete.
2010-12-14 00:32:39,539 INFO
org.apache.hadoop.hdfs.server.namenode.FSNamesystem:
Finished loading FSImage in 543 msecs
2010-12-14 00:32:39,544 INFO
```

```
org.apache.hadoop.hdfs.StateChange: STATE* Safe mode
ON.
The reported blocks 0 needs additional 9 blocks to
reach the threshold 0.9990 of total blocks 10. Safe
mode will be turned off automatically.
```

4. Notice a line that confirms the upgrade is complete, such as: **Upgrade of /var/lib/hadoop-0.20/cache/hadoop/dfs/name is complete.**
   If you use your web browser to access your NameNode on http://localhost:50070/, you should see a message: **Upgrade for version -19 has been completed. Upgrade is not finalized** appears. You will finalize the upgrade in a later step.

   Your NameNode is now in safe mode waiting for the DataNodes to connect.

Step 10: Start HDFS.

1. Start the rest of HDFS by running the following command on all the slave hosts:

```
$ sudo /etc/init.d/hadoop-0.20-datanode start
```

2. When the DataNodes start, they will connect to the NameNode and an upgrade will be initiated. You can monitor the upgrade from the NameNode console. After all of the DataNodes have upgraded, you should see a message **Safe mode will be turned off automatically in X seconds.**
3. After your DataNodes have completed the upgrade process, the NameNode should automatically exit safe mode and HDFS will be in full operation.
4. At this point, you can verify the operation of your HDFS instance by browsing your files, uploading new files, and so on.

Step 11: Change ownership of the mapred.system.dir directory on HDFS.

Because the JobTracker in CDH3 runs as a `mapred` user instead of a `hadoop` user, `mapred` must have ownership of the `mapred.system.dir` directory. After you start HDFS and before you start JobTracker, configure the owner of the `mapred.system.dir` directory to be the `mapred` user by using the local `hdfs` user as a superuser.

In the following command, the `mapred.system.dir` parameter is represented by the `/mapred/system` path example; change `/mapred/system` to match your

configuration. The default is `/var/lib/hadoop-0.20/cache/hadoop/mapred/system` if `mapred.system.dir` is not set.

```
$ sudo -u hdfs hadoop fs -chown mapred /mapred/system
```

Step 12: Start a Secondary NameNode and MapReduce.

1. Start a Secondary NameNode using its associated init.d script:

   ```
   $ sudo /etc/init.d/hadoop-0.20-secondarynamenode start
   ```

2. After you have verified HDFS is operating correctly, you are ready to start MapReduce. Because MapReduce does not keep a significant state persistence across upgrades, this process is much simpler.
   On the master host, run this command:

   ```
   $ sudo /etc/init.d/hadoop-0.20-jobtracker start
   ```

   On the slaves, run this command:

   ```
   $ sudo /etc/init.d/hadoop-0.20-tasktracker start
   ```

3. Verify that the Job Tracker and Task Tracker started properly.

   ```
   ps -eaf | grep -i job
   ps -eaf | grep -i task
   ```

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

Step 13: Verify basic cluster operation.

At this point your cluster is upgraded and ready to run jobs. Before running your production jobs, verify basic cluster operation by running a simple pi calculation job. You must first 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
```

Then run the pi calculation job as the user Joe:

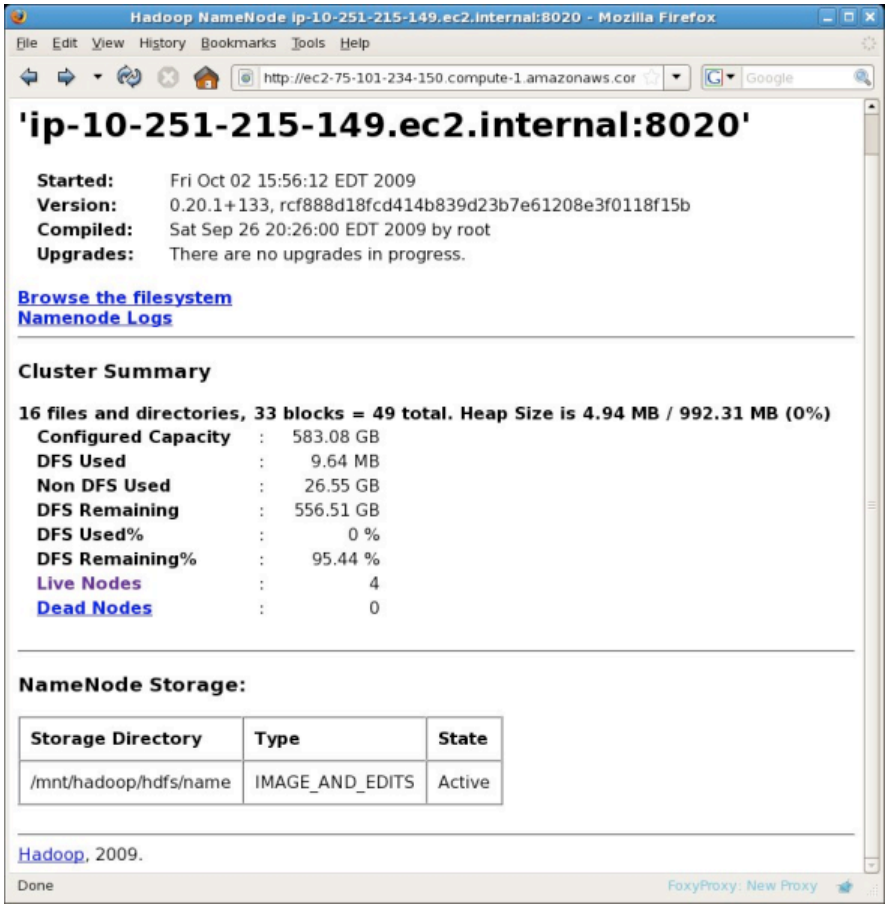
```
$ hadoop-0.20 jar /usr/lib/hadoop-0.20/hadoop-
0.20.2*examples.jar pi 10 10000
Number of Maps = 10
Samples per Map = 10000
...
Job Finished in 38.572 seconds
Estimated value of Pi is 3.14120000000000000000
```

Step 14: Finalize the HDFS upgrade.

You have now confirmed your cluster is successfully running on CDH3. Because you upgraded from a version prior to CDH3 Beta 3, you must finalize the HDFS upgrade by running this command:

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

After you refresh the NameNode console, you should see the message **Upgrades: There are no upgrades in progress.**



Important

If you have client hosts, make sure you also update them to CDH3, including the other CDH packages in the CDH distribution: Apache Pig, Apache Hive, Apache HBase, Flume, Sqoop, Hue, Oozie, Apache ZooKeeper, and Apache Whirr.

Step 15: (Optional) For security, set the sticky bit on directories.

This step is optional and strongly recommended for security. In CDH3, HDFS file permissions have support for the sticky bit. The sticky bit can be set on directories, preventing anyone except the superuser, directory owner, or file owner from deleting or moving the files within the directory. Setting the sticky bit for a file has no effect. This is useful for directories such as `/tmp` which previously had to be set to be world-writable. To set the sticky bit on the `/tmp` directory, run the following command:

```
$ sudo -u hdfs hadoop fs -chmod 1777 /tmp
```

After running this command, the permissions on `/tmp` will appear as shown below. (Note the "`t`" instead of the final "`x`".)

```
$ hadoop fs -ls /
Found 2 items
drwxrwxrwt - hdfs supergroup 0 2011-02-14 15:55 /tmp
drwxr-xr-x - hdfs supergroup 0 2011-02-14 14:01 /user
```

Configuring Ports for CDH3

The CDH3 components, and third parties such as Kerberos, use the ports listed in the tables that follow. Before you deploy CDH3, make sure these ports are open on each system.

Ports Used by Components of CDH3

Component	Service	Qualifier	Port	Protocol	Access Require-ment	Configuration	Comment
Flume	Master		57890	TCP		flume.master.admin.port	Admin/shell
	Master		45678	TCP		flume.master.gossip.port	Gossip
	Master		35872	TCP		flume.master.heartbeat.port	Heartbeat
	Master		35871	TCP		flume.master.http.port	Webserver
	ZooKeeper		3181	TCP		flume.master.zk.client.port	

Component	Service	Qualifier	Port	Protocol	Access Require-ment	Configuration	Comment
	Server						
	ZooKeeper Server		3183	TCP		flume.master.zk.server.election.port	
	ZooKeeper Server		3182	TCP		flume.master.zk.server.quorum.port	
	Node		35853	TCP		flume.collector.port	Collector
	Node		35862	TCP		flume.node.http.port	Status
Hadoop HDFS	DataNode		50010	TCP	External	dfs.datanode.address	xcievers
	DataNode	Secure	1004	TCP	External	dfs.datanode.address	xcievers
	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	
	DataNode	Thrift Plugin	0	TCP		dfs.thrift.datanode.address	
	NameNode		8020	TCP	External	fs.default.name	
	NameNode		50070	TCP	External	dfs.http.address	
	NameNode	Secure	50470	TCP	External	dfs.https.address	
	NameNode	Auth Plugin	10092	TCP	Internal	cloudera.plugin.namenode.port	
	NameNode	Thrift Plugin	10090	TCP		dfs.thrift.address	Required by Hue

Component	Service	Qualifier	Port	Protocol	Access Require-ment	Configuration	Comment
	Secondary NameNode		50090	TCP	Internal	dfs.secondary.http.address	
	Secondary NameNode	Secure	50495	TCP	Internal	dfs.secondary.https.address	
Hadoop MR	JobTracker		8021	TCP	External	mapred.job.tracker	
	JobTracker		50030	TCP	External	mapred.job.tracker.http.address	
	JobTracker	Auth Plugin	10091	TCP	Internal	cloudera.plugin.jobtracker.port	
	JobTracker	Thrift Plugin	9290	TCP	Internal	jobtracker.thrift.address	Required by Hue
	TaskTracke r		50060	TCP	External	mapred.task.tracker.http.address	
	TaskTracke r		0	TCP	Localhost	mapred.task.tracker.report.address	Communicati ng with child (umbilical)
HBase	Master		60000	TCP	External	hbase.master.port	IPC
	Master		60010	TCP	External	hbase.master.info.port	HTTP
	RegionServ er		60020	TCP	External	hbase.regionserver.port	IPC
	RegionServ er		60030	TCP	External	hbase.regionserver.info.port	HTTP
	HQuorumP eer		2888	TCP		hbase.zookeeper.property.clientPort	HBase-managed ZK mode

Component	Service	Qualifier	Port	Protocol	Access Require-ment	Configuration	Comment
	HQuorumPeer		3888	TCP		hbase.zookeeper.peerport	HBase-managed ZK mode
	HQuorumPeer		2181	TCP		hbase.zookeeper.leaderport	HBase-managed ZK mode
	REST	REST Service	8080	TCP	External	hbase.rest.port	
	ThriftServer	Thrift Server	9090	TCP	External	Pass `-p <port>` on CLI	
Hive	Metastore		9083	TCP	External		
	HiveServer		10000	TCP	External		
Hue	Server		8088	TCP	External		
	Job Submisison Server		8001		Internal		
	Beeswax Server		8002		Internal		
	Beeswax Metastore		8003		Internal		
Oozie	Server		11000	TCP	External		OOZIE_URL
	Server		8005	TCP	Internal	Hardcoded	Shutdown port
	Server	Auth	10094	TCP	Internal	cloudera.plugin.port	

Component	Service	Qualifier	Port	Protocol	Access Require-ment	Configuration	Comment
		Plugin					
Sqoop	Metastore		16000	TCP	External	sqoop.metastore.server.port	
ZooKeeper	Server		2181	TCP	External	clientPort	Client port
	Server		2888	TCP	Internal	X in server.N=host:X:Y	Peer
	Server		3888	TCP	Internal	Y in server.N=host:X:Y	Peer

Ports Used by Third Parties

Compone nt	Service	Qualifier	Port	Protocol	Access Require-ment	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 kdc.conf	By default only UDP
Kerberos	KRB5 Admin Server	Secure	749	TCP	Internal	kadmind_port in the [realms] section of kdc.conf	
LDAP							

CDH3 Deployment in Standalone Mode

Before you deploy CDH3 on a cluster, Cloudera recommends that you try deploying CDH3 on a local machine in standalone mode. In standalone mode, there is no distributed filesystem and no Java services/daemons are started. All mappers and reducers run inside a single Java VM.

To deploy CDH3 in standalone mode:

1. Verify that your package manager can list the Hadoop packages.

   List Hadoop packages on Ubuntu systems:

   ```
   $ apt-cache search hadoop
   ```

   List Hadoop packages on Red Hat systems:

   ```
   $ yum search hadoop
   ```

   List Hadoop packages on SUSE systems:

   ```
   $ zypper search hadoop
   ```

   You should see a list that includes packages such as `hadoop-0.20`, `hadoop-0.20-namenode`, `hadoop-0.20-datanode`, and so on. If you don't see any Hadoop packages, check your package manager setup.

2. Install Hadoop packages.

   On Ubuntu systems:

   ```
   $ sudo apt-get install hadoop-0.20
   ```

   On Red Hat systems:

   ```
   $ sudo yum install hadoop-0.20
   ```

   On SUSE systems:

   ```
   $ sudo zypper install hadoop-0.20
   ```

3. List the installed files.

   On Ubuntu systems:

```
$ dpkg -L hadoop-0.20
```

   On Red Hat and SUSE systems:

```
$ rpm -ql hadoop-0.20
```

   You can see that the Hadoop package has been configured to conform to the Linux Filesystem Hierarchy Standard. (To learn more, run `man hier`.).

```
Hadoop wrapper script          /usr/bin/hadoop-0.20
Hadoop Configuration Files     /etc/hadoop-0.20/conf
Hadoop Jar and Library Files   /usr/lib/hadoop-0.20
Hadoop Log Files               /var/log/hadoop-0.20
Hadoop Man pages               /usr/share/man
Hadoop service scripts         /etc/init.d/hadoop-
0.20-*
```

   To view the Hadoop man page:

```
$ man hadoop
```

4. You can run an example from the Apache Hadoop web site. First, you'll copy some XML files into a directory called input in the `/tmp` directory, and then use the XML files that are part of the Hadoop package you deployed.

   Copy some files into Hadoop for input:

```
$ cd /tmp
$ mkdir input
$ cp /etc/hadoop-0.20/conf/*.xml input
```

   Make sure there is no directory named `output20` in the `/tmp` directory. The following commands will start a job that creates and populates this

directory. The job will fail if the directory already exists.

5. Run the example grep job.

```
$ hadoop-0.20 jar /usr/lib/hadoop-0.20/hadoop-*-
examples.jar grep input output20 'dfs[a-z.]+'
```

After the job completes, the output for the job is written to the text file `/tmp/output20/part-00000`.

6. You can view the example grep job output.

```
$ cat /tmp/output20/part-00000 | head
 1       dfs.admin
```

CDH3 Deployment 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 CDH3 in pseudo-distributed mode:

1. Update your configuration to run in pseudo-distributed mode.

On Ubuntu systems:

```
$ sudo apt-get install hadoop-0.20-conf-pseudo
```

On Red Hat systems:

```
$ sudo yum install hadoop-0.20-conf-pseudo
```

On SUSE systems:

```
$ sudo zypper install hadoop-0.20-conf-pseudo
```

2. View how the `hadoop-0.20-conf-pseudo` packages changes your system.

   To view the files on Ubuntu systems:

   ```
   $ dpkg -L hadoop-0.20-conf-pseudo
   ```

   To view the files on Red Hat or SUSE systems:

   ```
   $ rpm -ql hadoop-0.20-conf-pseudo
   ```

   The new configuration is self-contained in the `/etc/hadoop-0.20/conf.pseudo` directory.

   Important

   The `hadoop-0.20-conf-pseudo` package automatically formats HDFS on installation if (*and only if*) the filesystem has not already been formatted previously. This HDFS formatting only initializes *files* in your `/var/lib/hadoop-0.20` directory and will not affect other filesystems on your machine.

   Note

   The Cloudera packages use the `alternative` framework for managing which Hadoop configuration is active. All Hadoop components search for the Hadoop configuration in `/etc/hadoop-0.20/conf`.

3. You need to start all the Hadoop services in order for the pseudo-distributed configuration to be functional. The `hadoop-0.20` package provides all of the scripts needed for managing Hadoop services. Start all services on Ubuntu, SUSE, and Red Hat systems by running these commands:

   ```
   $ for service in /etc/init.d/hadoop-0.20-*
   > do
   > sudo $service start
   > done
   ```

The NameNode provides a web console http://localhost:50070/ for viewing your Distributed File System (DFS) capacity, number of DataNodes, and logs. In the pseudo-distributed configuration, you should see one live DataNode named `localhost`.

4. The `JobTracker` provides a web console http://localhost:50030/ for viewing and running completed and failed jobs with logs. You can run an example from the Apache Hadoop web site. First, make a directory in HDFS called `input` and copy some xml files into it by running the following commands in pseudo-distributed mode:

```
$ hadoop-0.20 fs -mkdir input
$ hadoop-0.20 fs -put /etc/hadoop-0.20/conf/*.xml
input
$ hadoop-0.20 fs -ls input
Found 6 items
-rw-r--r-- 1 matt supergroup   6275  2009-08-18
18:36  /user/matt/input/capacity-scheduler.xml
-rw-r--r-- 1 matt supergroup    338  2009-08-18
18:36  /user/matt/input/core-site.xml
-rw-r--r-- 1 matt supergroup   3032  2009-08-18
18:36  /user/matt/input/fair-scheduler.xml
-rw-r--r-- 1 matt supergroup   4190  2009-08-18
18:36  /user/matt/input/hadoop-policy.xml
-rw-r--r-- 1 matt supergroup    496  2009-08-18
18:36  /user/matt/input/hdfs-site.xml
-rw-r--r-- 1 matt supergroup    213  2009-08-18
18:36  /user/matt/input/mapred-site.xml
```

5. Run an example Hadoop job to grep with a regex in your input data.

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

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

```
$ hadoop-0.20 fs -ls
Found 2 items
drwxr-xr-x   - matt supergroup  0 2009-08-18 18:36
```

```
/user/matt/input
drwxr-xr-x   - matt supergroup  0 2009-08-18 18:38
/user/matt/output
```

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

7. List the output files.

```
$ hadoop-0.20 fs -ls output
Found 2 items
drwxr-xr-x  -  matt supergroup     0 2009-02-25
10:33   /user/matt/output/_logs
-rw-r--r--  1  matt supergroup  1068 2009-02-25
10:33   /user/matt/output/part-00000
```

8. Read the results in the output file.

```
$ hadoop-0.20 fs -cat output/part-00000 | head
1   dfs.name.dir
1   dfs.permissions
1   dfs.replication
1   dfsadmin
```

CDH3 Deployment on a Cluster

Before you begin deploying CDH3 on a cluster, it's useful to review how Hadoop configurations work using the `alternatives` framework. The `alternatives` system is a set of commands used to manage symlinks to choose between multiple directories that fulfill the same purpose. CDH3 uses `alternatives` to manage your Hadoop configuration so that you can easily switch between cluster configurations. For more information on alternatives, see the `update-alternatives(8)` man page on Ubuntu systems or the `alternatives(8)` man page on Red Hat systems.

If you followed the deployment instructions in previous sections of this guide, you have two configurations: standalone and pseudo-distributed. To verify this, run the following commands.

To list alternative Hadoop configurations on Ubuntu and SUSE systems:

```
$ sudo update-alternatives --display hadoop-0.20-conf
hadoop-0.20-conf - status is auto.
link currently points to /etc/hadoop-0.20/conf.pseudo
/etc/hadoop-0.20/conf.empty - priority 10
/etc/hadoop-0.20/conf.pseudo - priority 30
Current `best' version is /etc/hadoop-0.20/conf.pseudo.
```

To list alternative Hadoop configurations on Red Hat systems:

```
$ sudo alternatives --display hadoop-0.20-conf
hadoop-0.20-conf - status is auto.
link currently points to /etc/hadoop-0.20/conf.pseudo
/etc/hadoop-0.20/conf.empty - priority 10
/etc/hadoop-0.20/conf.pseudo - priority 30
Current `best' version is /etc/hadoop-0.20/conf.pseudo.
```

Configuration

Customizing the Configuration without Using a Configuration Package

In the standalone and pseudo-distributed modes, the packages managed the configuration information for you. For example, when you installed the `hadoop-0.20-conf-pseudo` package, it added itself to the list of alternative `hadoop-0.20-conf` configurations with a priority of 30. This informed `alternatives` that the pseudo-distributed configuration should take precedence over the standalone configuration. You can customize the Hadoop configuration without using a configuration package.

To customize the Hadoop configuration without using a configuration package:

1. Copy the default configuration to your custom directory.

```
$ sudo cp -r  /etc/hadoop-0.20/conf.empty
/etc/hadoop-0.20/conf.my_cluster
```

You can call this configuration anything you like; in this example, it's called `my_cluster`. Edit the configuration files in `/etc/hadoop-0.20/conf.my_cluster` to suit your deployment.

2. Activate this new configuration by informing `alternatives` this configuration is a higher priority than the others (such as 50):

To activate the new configuration on Ubuntu and SUSE systems:

```
$ sudo update-alternatives --install /etc/hadoop-
0.20/conf hadoop-0.20-conf /etc/hadoop-
0.20/conf.my_cluster 50
```

To activate the new configuration on Red Hat systems:

```
$ sudo alternatives --install /etc/hadoop-0.20/conf
hadoop-0.20-conf /etc/hadoop-0.20/conf.my_cluster 50
```

3. If you look at the `/etc/hadoop-0.20/conf` symlink now, it points to `/etc/hadoop-0.20/conf.my_cluster`. You can verify this by querying the `alternatives`.

To query Hadoop configuration alternatives on Ubuntu and SUSE systems:

```
$ sudo update-alternatives --display hadoop-0.20-
conf
hadoop-0.20-conf - status is auto.
link currently points to /etc/hadoop-
0.20/conf.my_cluster
/etc/hadoop-0.20/conf.empty - priority 10
/etc/hadoop-0.20/conf.pseudo - priority 30
/etc/hadoop-0.20/conf.my_cluster - priority 50
Current `best' version is /etc/hadoop-
0.20/conf.my_cluster.
```

To query Hadoop configuration alternatives on Red Hat systems:

```
$ sudo alternatives --display hadoop-0.20-conf
hadoop-0.20-conf - status is auto.
link currently points to /etc/hadoop-
0.20/conf.my_cluster
/etc/hadoop-0.20/conf.empty - priority 10
/etc/hadoop-0.20/conf.pseudo - priority 30
/etc/hadoop-0.20/conf.my_cluster - priority 50
Current `best' version is /etc/hadoop-
0.20/conf.my_cluster.
```

The `alternatives` system knows of three Hadoop configurations and has chosen your new `conf.my_cluster` configuration because it has a higher priority.

> If you install two configurations with the same priority, `alternatives` will choose the one installed last.

4. You can remove your configuration from the list of alternatives.
   To remove alternative configuration on Ubuntu and SUSE systems:

```
$ sudo update-alternatives --remove hadoop-0.20-conf
/etc/hadoop-0.20/conf.my_cluster
```

   To remove alternative configuration on Red Hat systems:

```
$ sudo alternatives --remove hadoop-0.20-conf
/etc/hadoop-0.20/conf.my_cluster
```

5. You can tell alternatives to manually choose a configuration without regard to priority.
   To manually set the configuration on Ubuntu and SUSE systems:

```
$ sudo update-alternatives --set hadoop-0.20-conf
/etc/hadoop-0.20/conf.my_cluster
```

To manually set the configuration on Red Hat systems:

```
$ sudo alternatives --set hadoop-0.20-conf
/etc/hadoop-0.20/conf.my_cluster
```

Configuration Files and Properties

The following table shows the most important HDFS properties that you must configure.

Property	Configuration File	Description
`fs.default.name`	`conf/core-site.xml`	Specifies the NameNode and the default file system, in the form `hdfs://<namenode>/`. The default value is `file///`. The default file system is used to resolve relative paths; for example, if `fs.default.name` is set to `hdfs://mynamenode/`, the relative path `/mydir/myfile` resolves to `hdfs://mynamenode/mydir/myfile`.
`mapred.job.tracker`	`conf/mapred-site.xml`	Specifies the hostname and port of the JobTracker's RPC server, in the form `<host>:<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**
>
> For information on other important configuration properties, and the configuration files, see the Apache Cluster Setup page.

Configuring Local Storage Directories for Use by HDFS and MapReduce

When moving from a pseudo distributed cluster to a full cluster deployment, you will need to specify, create, and assign the correct permissions to the local directories where you want the HDFS and MapReduce daemons to store data.

You specify the directories by configuring the following three properties; two properties are in the `hdfs-site.xml` file, and one property is in the `mapred-site.xml` file:

Property	Configuration File Location	Description
`dfs.name.dir`	`hdfs-site.xml` on the NameNode	This property specifies the directories where the NameNode stores its metadata and edit logs. Cloudera recommends that you specify at least two directories, one of which is located on an NFS mount point.
`dfs.data.dir`	`hdfs-site.xml` on each DataNode	This property specifies 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` to specify `/data/1/dfs/dn` through `/data/N/dfs/dn/`.

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`.

Here is an example configuration:

hdfs-site.xml:

```
<property>
 <name>dfs.name.dir</name>
 <value>/data/1/dfs/nn,/nfsmount/dfs/nn</value>
</property>
<property>
 <name>dfs.data.dir</name>

<value>/data/1/dfs/dn,/data/2/dfs/dn,/data/3/dfs/dn</valu
e>
</property>
```

mapred-site.xml:

```
<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` and `hdfs-site.xml` files, 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` parameter is represented by the `/data/1/dfs/nn` and `/nfsmount/dfs/nn` path examples.
- The `dfs.data.dir` parameter is represented by the `/data/1/dfs/dn`, `/data/2/dfs/dn`, `/data/3/dfs/dn`, and `/data/4/dfs/dn` path examples.
- 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.

To configure local storage directories for use by HDFS and MapReduce:

1. Create the `dfs.name.dir` local directories:

```
$ sudo mkdir -p /data/1/dfs/nn /nfsmount/dfs/nn
```

2. Create the `dfs.data.dir` local directories:

```
$ sudo mkdir -p /data/1/dfs/dn /data/2/dfs/dn
/data/3/dfs/dn /data/4/dfs/dn
```

3. 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
```

4. Configure the owner of the `dfs.name.dir` and `dfs.data.dir` directories to be the `hdfs` user:

```
$ sudo chown -R hdfs:hadoop /data/1/dfs/nn
/nfsmount/dfs/nn /data/1/dfs/dn /data/2/dfs/dn
/data/3/dfs/dn /data/4/dfs/dn
```

5. 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
```

Here is a summary of the correct owner and permissions of the local directories:

Directory	**Owner**	**Permissions (see Footnote [1])**
`dfs.name.dir`	`hdfs:hadoop`	drwx------
`dfs.data.dir`	`hdfs:hadoop`	drwx------
`mapred.local.dir`	`mapred:hadoop`	drwxr-xr-x

Footnote:
[1] In CDH3, the Hadoop daemons automatically set the correct permissions for you on the `dfs.data.dir` and `mapred.local.dir` directories if you configure the directory ownership correctly as shown in the table above. But in the case of `dfs.name.dir`, permissions are currently (in releases through CDH3 Update 2) incorrectly set to the file-system default, usually drwxr-xr-x (755). Use the `chmod` command to reset permissions for these `dfs.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` property in the `conf/hdfs-site.xml` file, CDH3 will shut down. (In previous releases, CDH3 silently ignored nonexistent directories for `dfs.data.dir`.)

Creating and Configuring the mapred.system.dir Directory in HDFS

1. After you start HDFS and before you start JobTracker, you must also create the HDFS directory specified by the `mapred.system.dir` parameter and configure it to be owned by the `mapred` user. The `mapred.system.dir` parameter is represented by the following `/mapred/system` path example.

```
$ sudo -u hdfs hadoop fs -mkdir /mapred/system
$ sudo -u hdfs hadoop fs -chown mapred:hadoop
/mapred/system
```

   Here is a summary of the correct owner and permissions of the `mapred.system.dir` directory in HDFS:

Directory	Owner	Permissions
`mapred.system.dir`	`mapred:hadoop`	drwx------ [1]
/ (root directory)	`hdfs:hadoop`	drwxr-xr-x

   Footnote:
   [1] When starting up, MapReduce sets the permissions for the `mapred.system.dir` directory in HDFS, assuming the user `mapred` owns that directory.

2. Add the path for the `mapred.system.dir` directory to the `conf/mapred-site.xml` file.

Deploying your Custom Configuration to your Entire Cluster

To deploy your custom configuration to your entire cluster:

1. Push the `/etc/hadoop-0.20/conf.my_cluster` directory to all nodes in your cluster.
2. Set `alternative` rules on all nodes to activate your configuration.
3. Restart the daemons on all nodes in your cluster using the service scripts so that the new configuration files are read.

Initial Setup

Format the NameNode

Before starting the NameNode for the first time you need to format the file system.

> Make sure you format the NameNode as user `hdfs`.

```
$ sudo -u hdfs hadoop namenode -format
```

Configuring Backup for the NameNode

You should configure the NameNode to write to multiple disks, including at least one 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.

For 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.

For more information, see the man pages for `mount` and `nfs`.

Running Services

Starting HDFS

To start HDFS:

```
$ sudo service hadoop-0.20-namenode start
$ sudo service hadoop-0.20-secondarynamenode start
$ sudo service hadoop-0.20-datanode start
```

Creating the HDFS /tmp Directory

Once HDFS is up and running, create the /tmp directory and set its permissions to 1777 (drwxrwxrwt), as follows:

```
$ sudo -u hdfs hadoop dfs -mkdir /tmp
$ sudo -u hdfs hadoop dfs -chmod -R 1777 /tmp
```

Starting MapReduce

To start MapReduce:

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

Configuring the Hadoop Daemons to Start at Boot Time

To start the Hadoop daemons at boot time and on restarts, enable their `init` scripts using the `chkconfig` tool:

```
$ sudo chkconfig hadoop-0.20-namenode on
$ sudo chkconfig hadoop-0.20-jobtracker on
$ sudo chkconfig hadoop-0.20-secondarynamenode on
$ sudo chkconfig hadoop-0.20-tasktracker on
$ sudo chkconfig hadoop-0.20-datanode on
```

On Ubuntu systems, you can install the `sysv-rc-conf` package to get the `chkconfig` command or use `update-rc.d`:

```
$ sudo update-rc.d hadoop-0.20-namenode defaults
$ sudo update-rc.d hadoop-0.20-jobtracker defaults
$ sudo update-rc.d hadoop-0.20-secondarynamenode defaults
$ sudo update-rc.d hadoop-0.20-tasktracker defaults
$ sudo update-rc.d hadoop-0.20-datanode defaults
```

Note that you must run the commands on the correct server, according to your role definitions.

> **Note**
>
> For more information about configuring Hadoop on a cluster, see Cluster Setup.

When SSH is and is not Used

It is a good idea to use SSH for remote administration purposes (instead of `rlogin`, for example) but note that Hadoop itself and the related services do not use SSH for communication as a matter of course. Some scripts, and in particular the Hadoop `start-all` and `stop-all` scripts, do use SSH, but otherwise SSH is not used for communication among the following:

- Datanode
- Namenode
- TaskTracker
- JobTracker
- `/etc/init.d` scripts (which start daemons locally)

Flume Installation

Flume is a distributed, reliable, and available service for efficiently moving large amounts of data as the data is produced. This release provides a scalable conduit to shipping data around a cluster and concentrates on reliable logging. The primary use case is as a logging system that gathers a set of log files on every machine in a cluster and aggregates them to a centralized persistent store such as HDFS.

Upgrading Flume in CDH3

If you used Flume in versions prior to CDH3 Update 1, read this section. Flume version 0.9.4 in CDH3u1 includes new properties in the `flume-conf.xml` file. If you are

upgrading to CDH 3u1, then you must do *one* of the following to prevent failures from occurring in Flume's internal web applications:

- Add the following new properties to your existing Flume 0.9.3 `flume-conf.xml` configuration file:
  - `flume.node.webapp.root`
  - `flume.master.webapp.root`
  - `flume.node.wal.output`
    To conform with the defaults, set the `flume.node.webapp.root` property to point to `webapps/flumeagent.war`; set `flume.master.webapp.root` to point to `webapps/flumemaster.war`; and set `flume.node.wal.output` to `true` (this value will make the WAL subsystem buffer writes to disk, improving performance at the cost of durability).

- Alternatively, if you are using the default settings in your existing Flume 0.9.3 `flume-conf.xml` configuration file, you can overwrite it with the new Flume 0.9.4 `flume-conf.xml` file.

Upgrading Flume to CDH3 Update 2

The instructions that follow assume that you are upgrading Flume as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

To upgrade Flume to CDH 3 Update 2, proceed as follows:

Step 1: Stop the Flume Node Processes on each Node where They are Running

To stop the Flume node process:

```
/etc/init.d/flume-node stop
```

Step 2: Stop the Flume Master

```
/etc/init.d/flume-master stop
```

Step 3. Install the new version of Flume

Proceed as follows:

1. Make sure you understand the packaging options and the prerequisites.
2. Do one of the following:

- Install Flume from a tarball; see Installing the Flume Tarball.
  or
- Install Flume RPM or Debian Packages; see Installing the Flume RPM or Debian Packages.

Step 4. Start the Flume Nodes and Flume Master

1. Start the Flume nodes; see Starting Flume Nodes on Boot Up Automatically.
2. Start the Flume master; see Starting the Flume Masters on Boot Up Automatically.

The upgrade is now complete. For more information, see Running Flume, Files Installed by the Flume RPM and Debian Packages, and Viewing the Flume Documentation.

Flume Packaging

There are currently three packaging options available for installing Flume:

- Tarball (.tgz)
- RPM packages
- Debian packages
- Microsoft Windows Executable Installer (for Flume Node only)

> **Note**
>
> For instructions on how to install Flume nodes on Windows, see Installing Flume Nodes on Windows.

Flume Prerequisites

- A Unix-like system (tested on Centos 5.5+, SLES11 and Ubuntu 9.04+)
- Java 1.6.x (only tested with JRE/JDK 1.6)

-

Important

Before you can run Flume Master, the Hadoop and Apache Zookeeper packages must be installed. Although both the Hadoop and Zookeeper packages must be installed, neither need to be running as services on the target machines in order for Flume to run. For installation instructions, see CDH3 Installation and ZooKeeper Installation.

Installing the Flume Tarball (.tgz)

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 command:

```
$ (cd /usr/local/ && sudo tar -zxvf
<path_to_flume.tgz>)
```

2. To complete the configuration of a tarball installation, you must set the environment variable `$FLUME_CONF_DIR` to be the `conf/` subdirectory of the directory where you installed Flume. For example:

```
$ export FLUME_CONF_DIR=/usr/local/flume/conf
```

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` — Everything you need to run Flume
- `flume-master` — Handles starting and stopping the Flume master as a service
- `flume-node` — Handles starting and stopping the Flume node as a service

All Flume installations require the common code provided by `flume`.

> **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 CDH3 Installation.

To install Flume on Ubuntu and other Debian systems:

```
$ sudo apt-get install flume
```

To install Flume on Red Hat systems:

```
$ sudo yum install flume
```

To install Flume on SUSE systems:

```
$ sudo zypper install flume
```

At this point, you should have everything necessary to run Flume, and the `flume` command should be in your `$PATH`. You can test this by running:

```
$ flume
```

You should see something similar to this:

```
usage: flume command [args...]
commands include:
  dump             Takes a specified source and dumps to
console
  node             Start a Flume node/agent (with
watchdog)
  master           Start a Flume Master server (with
watchdog)
  version          Dump flume build version information
  node_nowatch     Start a flume node/agent (no watchdog)
  master_nowatch   Start a Flume Master server (no
watchdog)
  class <class>    Run specified fully qualified class
using Flume environment (no watchdog)
                    for example: flume
com.cloudera.flume.agent.FlumeNode
  shell            Start the flume shell
  killmaster       Kill a running master
```

> **Note**
>
> If Flume is not found and you installed Flume from a tarball, make sure that `$FLUME_HOME/bin` is in your `$PATH`.

The final step in a package-based installation is to automatically start Flume via system runtime levels. The `flume-node` and `flume-master` packages make this easy.

Starting Flume Nodes on Boot Up Automatically

To start Flume nodes automatically on boot on an Ubuntu or other Debian system:

```
$ sudo apt-get install flume-node
```

To start Flume nodes automatically on boot on a Red Hat system:

```
$ sudo yum install flume-node
```

To start Flume nodes automatically on boot on a SUSE system:

```
$ sudo zypper install flume-node
```

Starting the Flume Master on Boot Up Automatically

To start the Flume master automatically on boot on a Debian system:

```
$ sudo apt-get install flume-master
```

To start the Flume master automatically on boot on a Red Hat system:

```
$ sudo yum install flume-master
```

To start the Flume master automatically on boot on a SUSE system:

```
$ sudo zypper install flume-master
```

Running Flume

If Flume is installed via a RPM or Debian package, you can use the following commands to start, stop, and restart the Flume master and the Flume node via `init` scripts:

To start, stop, or restart the Flume master:

```
$ /etc/init.d/flume-master <start | stop | restart>
```

To start, stop, or restart the Flume node:

```
$ /etc/init.d/flume-node <start | stop | restart>
```

You can also run the node and master in the foreground directly by using the `flume` command with any of the installation methods.

To start the Flume node:

```
$ flume node
```

To start the Flume master:

```
$ flume master
```

> **Note**
>
> When using Flume for the first time, it's common to try running both master and node on the same machine. This should work without a problem — both node and master can share the same configuration files and should not conflict on any ports they open. See the **Pseudo-Distributed Mode** section in the **Flume User Guide** for more details.

Files Installed by the Flume RPM and Debian Packages

Resource	Location
Config Directory	`/etc/flume/conf`
Template of User Customizable Config File [1]	`/etc/flume/conf/flume-site.template.xml`

Resource	Location
Daemon Log Directory	`/var/log/flume`
Default Flume Home [2]	`/usr/lib/flume`
Flume Master startup script ^2	`/etc/init.d/flume-master`
Flume Node startup script [2]	`/etc/init.d/flume-node`
Recommended TGZ Flume Home [3]	`/usr/local/lib/flume`
Flume Wrapper Script	`/usr/bin/flume`

Footnotes:

[1] You should copy this file, modify it as necessary, and rename it to `flume-site.xml`. For example:

```
cd /etc/flume/conf
cp flume-site.template.xml flume-site.xml
```

Flume looks for `flume-site.xml` at runtime.

[2] Provided by RPMS and DEBS

[3] Recommended but installation dependent

Viewing the Flume Documentation

For additional Flume documentation, see:

- Flume User Guide
- Flume Cookbook
- Installing Flume Nodes on Windows
- Flume FAQ

For additional information about Flume, see the Flume github wiki.

Sqoop Installation

Apache Sqoop is a tool designed for efficiently transferring bulk data between Apache Hadoop and structured datastores such as relational databases. You can use Sqoop 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 to extract data from Hadoop and export it to external structured datastores such as relational databases and enterprise data warehouses.

Upgrading Sqoop to CDH3 Update 2

The instructions that follow assume that you are upgrading Sqoop as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

> **Note**
>
> Because of a packaging problem in earlier releases, you need stop the Sqoop metastore service before using RPM or Debian packages to upgrade Sqoop to CDH3 Update 2, then re-enable it manually after the upgrade, as described below. This applies only to upgrading Sqoop to CDH3 Update 2, and does not apply if you are upgrading from a tarball.

To upgrade Sqoop to CDH3 Update 2, proceed as follows:

1. If you are using RPM or Debian packages, stop the Sqoop metastore service:

```
$ sudo service sqoop-metastore stop
```

2. Install the new version of Sqoop using one of the methods described below: Installing the Sqoop RPM or Debian Packages or Installing the Sqoop Tarball.
3. If you are using RPM or Debian packages, re-enable the Sqoop metastore:

```
$ sudo /sbin/chkconfig --add sqoop-metastore
```

4. If you are using RPM or Debian packages, restart the Sqoop metastore service:

```
$ sudo service sqoop-metastore start
```

The upgrade is now complete.

Sqoop Packaging

There are currently three packaging options available for installing Sqoop:

- RPM packages
- Debian packages
- Tarball

Sqoop Prerequisites

- A Unix-like system (tested on Centos 5.5+, SLES11 and Ubuntu 9.04+)
- Java 1.6.x (only tested with JDK 1.6)

Installing the Sqoop RPM or Debian Packages

Installing the Sqoop RPM or Debian packages is more convenient than installing the Sqoop tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations

The Sqoop RPM and Debian packages consist of two packages:

- `sqoop` — Complete Sqoop distribution
- `sqoop-metastore` — For installation of the Sqoop metastore only

> **Important**
>
> If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` respository before using the following commands to install Sqoop. For instructions, see CDH3 Installation.

To install Sqoop on an Ubuntu or other Debian system:

```
$ sudo apt-get install sqoop
```

To install Sqoop on a Red Hat system:

```
$ sudo yum install sqoop
```

To install Sqoop on a SUSE system:

```
$ sudo zypper install sqoop
```

If you have already configured CDH on your system, then there is no further configuration necessary for Sqoop. You can start using Sqoop by using commands such as:

```
$ sqoop help
$ sqoop version
$ sqoop import
```

Installing the Sqoop Tarball

The Sqoop tarball is a self-contained package containing everything necessary to use Sqoop on a Unix-like system. To install Sqoop 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 commands. Documentation for users and developers can be found in the `docs` directory.

To install the Sqoop tarball on Linux-based systems:

Run the following command:

```
$ (cd /usr/local/ && sudo tar \-zxvf
_<path_to_sqoop.tar.gz>_)
```

> **Note**
>
> When installing Sqoop from the tarball package, you must make sure that the environment variables `JAVA_HOME` and `HADOOP_HOME` are configured correctly. The variable `HADOOP_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.

Viewing the Sqoop Documentation

For additional documentation see the Sqoop User Guide and the Sqoop Developer Guide.

Hue Installation

Hue is a graphical user interface to work with CDH. Hue aggregates several applications which are collected into a desktop-like environment and delivered as a Web application that requires no client installation by individual users.

> **Important**
>
> If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` respository before using the following commands to install Hue. For instructions, see CDH3 Installation.

If you are installing Hue on a cluster with CDH in distributed mode, skip to Installing and Configuring Hue on a Cluster for instructions.

Upgrading Hue in CDH3

If you used the embedded Hive MetaStore functionality of Beeswax in Hue in versions prior to CDH3 Beta 4, read this section. This CDH3 release includes changes in the Hive MetaStore schema that are part of the Hive 0.7 release. If you want to use Beeswax in Hue in CDH3, it is imperative that you upgrade the Hive MetaStore schema by running the appropriate schema upgrade script located in the `/usr/lib/hive/scripts/metastore/upgrade` directory. Scripts for Derby

and MySQL databases are available. If you are using a different database for your MetaStore, you will need to provide your own upgrade script.

The following table summarizes the versions of Hive that each Hue version uses:

This Hue Version	Uses this Hive Version
1.1.x	Hue bundles 0.5
1.2.0	Hue bundles 0.7 release candidate 0
1.2.0.0+x [1] (CDH3 Update 1)	Hue no longer bundles Hive. Hue requires at least Hive 0.7.0 also be installed.

Footnote:
[1] For the Hue 1.2.0+x patch level number, see Downloading CDH Releases.

For more information about upgrading to Hive 0.7 and the Hive MetaStore functionality, see Hive Installation.

Upgrading Hue from CDH3 Beta 4 to CDH3 Update 1 or Later

If you upgraded Hue from CDH3 Beta 4 to CDH3 Update 1 or later, and use the embedded metastore feature of Beeswax, the default metastore location has changed. The result of this is that your tables will no longer be listed in Beeswax. You can fix this problem by editing the metastore location specified in the `hive-site.xml` file as described below.

1. Open the `/etc/hive/conf/hive-site.xml` file.
2. Change the location of the tables by changing the value of the `javax.jdo.option.ConnectionURL` property:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>

<value>jdbc:derby:;databaseName=/usr/share/hue/metas
tore_db;create=true</value>
  <description>JDBC connect string for a JDBC
metastore</description>
</property>
```

Upgrading Hue from CDH3 Update 1 to CDH3 Update 2

You can upgrade Hue either as part of an overall upgrade to CDH3 Upgrade 2 (see Upgrading to CDH3) or independently.

To upgrade Hue from CDH3 Update 1 to CDH 3 Update 2, proceed as follows.

Step 1: Stop Hue on the Hue Machine

To stop Hue:

```
sudo /etc/init.d/hue stop
```

Step 2: Install the new version of Hue

See Installing Hue on a Cluster.

Step 3: Restart Hue

To restart Hue:

```
sudo /etc/init.d/hue start
```

The upgrade is now complete.

Installing, Configuring, and Starting Hue on One Machine

If you're installing Hue on one machine with CDH in pseudo-distributed mode, you can install Hue with only one command and setting one configuration setting as shown in this section.

To install Hue on Ubuntu and other Debian systems:

```
$ sudo apt-get install hue
```

To install Hue on Red Hat-based systems:

```
$ sudo yum install hue
```

To install Hue on SUSE-based systems:

```
$ sudo zypper install hue
```

Important

The Beeswax server writes into a local directory on the Hue machine that is specified by `hadoop.tmp.dir` to unpack its jars. That directory needs to be writable by the `hue` user, which is the default user who starts Beeswax Server, or else Beeswax server will not start. You may also make that directory world-writable. For more information about `hadoop.tmp.dir`, see the "hadoop.tmp.dir" section in the Hue manual.

Specifying the Secret Key

For security, you should also specify the secret key that is used for secure hashing in the session store:

1. Open the `/etc/hue/hue.ini` configuration file.

   Note

   Only the root user can edit this file.

2. In the [desktop] section, enter a long series of random characters (30 to 60 characters is recommended).

   ```
   [desktop]
   secret_key=jFE93j;2[290-
   eiw.KEiwN2s3['d;/.q[eIW^y#e=+Iei*@Mn<qW5o
   ```

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.

Starting Hue on One Machine

To start and use Hue on one machine:

1. Start the Hadoop daemons in pseudo-distributed mode.

```
$ for service in /etc/init.d/hadoop-0.20-*; do sudo
$service start; done
```

2. Start Hue by running:

```
$ sudo /etc/init.d/hue start
```

3. To use Hue, open a web browser and go to: http://localhost:8088/

Installing and Configuring Hue on a Cluster

This section describes Hue installation and configuration on a cluster. Hue ships with a default configuration that works for pseudo-distributed clusters. If you are running on a real cluster, you must make a few configuration changes.

Installing Hue on a Cluster

You must install the `hue-common` package on the machine where you will run Hue. In addition, you must install the `hue-plugins` package on every machine in your CDH cluster.

On Ubuntu and other Debian Systems:

To install the `hue-common` package and all Hue applications on the Hue machine on Ubuntu and other Debian systems, use the `hue` meta-package:

```
$ sudo apt-get install hue
```

To install the `hue-plugins` package on every CDH machine:

```
$ sudo apt-get install hue-plugins
```

On Red Hat Systems:

To install the `hue-common` package and all Hue applications on the Hue machine on Red Hat systems, use the `hue` meta-package:

```
$ sudo yum install hue
```

To install the `hue-plugins` package on every CDH machine:

```
$ sudo yum install hue-plugins
```

On SUSE Systems:

To install the `hue-common` package and all Hue applications on the Hue machine on SUSE systems, use the `hue` meta-package:

```
$ sudo zypper install hue
```

To install the `hue-plugins` package on every CDH machine:

```
$ sudo zypper install hue-plugins
```

> **Important**
>
> The Beeswax server writes into a local directory on the Hue machine that is specified by `hadoop.tmp.dir` to unpack its jars. That directory needs to be writable by the `hue` user, which is the default user who starts Beeswax Server, or else Beeswax server will not start. You may also make that directory world-writable. For more information about `hadoop.tmp.dir`, see the Hue manual.

Specifying the Secret Key

For security, you should also specify the secret key that is used for secure hashing in the session store:

1. Open the `/etc/hue/hue.ini` configuration file.

> **Note**
>
> Only the root user can edit this file.

2. In the [desktop] section, enter a long series of random characters (30 to 60 characters is recommended).

```
[desktop]
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.

Configuring the Hadoop Plugins for Hue

To enable communication between Hue and CDH, you must make minor changes to your CDH installation by adding the following properties to your CDH configuration files in `/etc/hadoop-0.20/conf/`. Make these configuration changes on each node in your cluster.

Add the following properties to the hdfs-site.xml :

```
<property>
  <name>dfs.namenode.plugins</name>
  <value>org.apache.hadoop.thriftfs.NamenodePlugin</value>
  <description>Comma-separated list of namenode plug-ins to
be activated.
  </description>
</property>
<property>
  <name>dfs.datanode.plugins</name>
  <value>org.apache.hadoop.thriftfs.DatanodePlugin</value>
  <description>Comma-separated list of datanode plug-ins to
be activated.
  </description>
</property>
```

```
<property>
  <name>dfs.thrift.address</name>
  <value>0.0.0.0:10090</value>
</property>
```

Add the following properties to the mapred-site.xml file:

```
<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>
```

Restarting the Hadoop Daemons

After you have installed Hue and configured the Hadoop plugins, you must restart the Hadoop daemons.

1. On the master node, run:

```
$ sudo service hadoop-0.20-namenode restart
$ sudo service hadoop-0.20-jobtracker restart
```

2. On all the slave nodes, run:

```
$ sudo service hadoop-0.20-datanode restart
```

Pointing Hue to Your CDH NameNode and JobTracker

After you have restarted the Hadoop daemons, you must configure Hue to point to the external hostnames of your NameNode and JobTracker. To do so, change the

`namenode_host` and `jobtracker_host` lines in the `/etc/hue/hue.ini` configuration file.

To point Hue to your CDH NameNode and JobTracker:

1. If you installed Hadoop in a different location, you must specify the path by configuring `hadoop_home`, which is where `bin/hadoop`, the Hadoop wrapper script, is found:

```
[hadoop]
hadoop_home=/usr/lib/hadoop-0.20
```

2. Specify the host and port on which you are running the Hadoop NameNode by configuring `namenode_host` and `hdfs_port`:

```
# Configuration for HDFS NameNode
# ----------------------------------------------------
----------------------
[[hdfs_clusters]]
[[[default]]]
# Enter the host and port on which you are running
the Hadoop NameNode
namenode_host=localhost
hdfs_port=8020
# Thrift plugin port for the name node
## thrift_port=10090
```

3. Specify the host on which you are running the MapReduce JobTracker by configuring `jobtracker_host`:

```
# Configuration for MapReduce JobTracker
# ----------------------------------------------------
----------------------
[[mapred_clusters]]
[[[default]]]
# Enter the host on which you are running the Hadoop
JobTracker
jobtracker_host=localhost
# Thrift plug-in port for the JobTracker
## thrift_port=9290
```

Web Server Configuration

Starting with CDH3 Update 1, Hue includes two web servers, the CherryPy web server and the Spawning 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 8088 on all configured IP addresses.

```
# Webserver listens on this address and port
http_host=0.0.0.0
http_port=8088
```

Starting with CDH3 Update 1, Hue defaults to using the Spawning server, which is necessary for the Shell application, a new feature in CDH3 Update 1. To revert to the CherryPy server, use the following setting in the `/etc/hue/hue.ini` configuration file:

```
use_cherrypy_server=true
```

Setting this to false also uses the Spawning web server.

Authentication

By default, the first user who logs in to Hue can choose any username and password and becomes an administrator automatically. This user can create other user and administrator accounts. User information is stored in the Django database in the Django backend.

The authentication system is pluggable. For more information, see the Hue SDK.

Configuring Hue for SSL

You can optionally configure Hue to serve over HTTPS. To do so, you must install pyOpenSSL within Hue's context and configure your keys.

To install pyOpenSSL, do the following steps from the root of your Hue installation path:

1. Run this command:

```
$ ./build/env/bin/easy_install pyOpenSSL
```

2. Configure Hue to use your private key by adding the following options to the `/etc/hue/hue.ini` configuration file:

```
ssl_certificate=/path/to/certificate
ssl_private_key=/path/to/key
```

3. Ideally, you would have an appropriate key signed by a 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
```

> **Self-signed Certificates and File Uploads**
>
> To upload files using the Hue File Browser over HTTPS requires using a proper SSL Certificate. Self-signed certificates don't work.

Listing all Configuration Options

To list all available configuration options, run:

```
$ /usr/share/hue/build/env/bin/hue config_help | less
```

This command outlines the various sections and options in the configuration, and provides help and information on the default values.

Viewing Current Configuration Settings

To view the current configuration settings from within Hue, open:

```
http://<hue-host:hue-port>/dump_config
```

where:

`<hue-host:hue-port>` is localhost or the IP address of the Hue machine and the Hue port (the default is 8088). For example: http://localhost:8088/

Using Multiple Files to Store Your Configuration

Hue loads and merges all of the files with extension `.ini` located in the `/etc/hue/conf/` directory. Files that are alphabetically later take precedence.

Starting and Stopping Hue

The `hue-common` package include service scripts to start and stop Hue.

To start Hue:

```
$ sudo /etc/init.d/hue start
```

To stop Hue:

```
$ sudo /etc/init.d/hue stop
```

> **Note**
>
> Hue includes several processes; a `supervisor` daemon manages the web server along with some helper processes.

Hue Process Hierarchy

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 following processes:

- `runcpserver` – a web server based on CherryPy that provides the core web functionality of Hue
- `jobsubd` – a daemon which handles submission of jobs to Hadoop
- `beeswax server` – a daemon that manages concurrent Hive queries

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`:

```
UID        PID  PPID  C STIME TTY          TIME CMD
hue       8685  8679  0 Aug05 ?        00:01:39
/usr/share/hue/build/env/bin/python
/usr/share/hue/build/env/bin/desktop runcpserver
hue       8693  8679  0 Aug05 ?        00:00:01
/usr/share/hue/build/env/bin/python
/usr/share/hue/build/env/bin/desktop jobsubd
hue       8695  8679  0 Aug05 ?        00:00:06
/usr/java/jdk1.6.0_14/bin/java -Xmx1000m -
Dhadoop.log.dir=/usr/lib/hadoop-0.20/logs -
Dhadoop.log.file=hadoop.log ...
```

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.

Hue Logging

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 web 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. If you are unable to start Hue from the init script, the `supervisor.log` file can often contain clues.

Viewing Recent Log Messages through your Web Browser

In addition to logging `INFO` level messages to the logs directory, the Hue web server keeps a small buffer of log messages at all levels in memory. You can view these logs by

visiting `http://myserver:8088/logs`. The `DEBUG` level messages shown can sometimes be helpful in troubleshooting issues.

The Hue Database

Hue requires a SQL database to store small amounts of data, including user account information as well as history of job submissions and Hive queries. By default, Hue is configured to use the embedded database SQLite for this purpose, and should require no configuration or management by the administrator. However, MySQL is the recommended database to use; this section contains instructions for configuring Hue to access MySQL and other databases.

Inspecting the Hue Database

The default SQLite database used by Hue is located in `/usr/share/hue/desktop/desktop.db`. You can inspect this database from the command line using the `sqlite3` program. For example:

```
# sqlite3 /usr/share/hue/desktop/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 SQLite, though this trick can be useful for management or troubleshooting.

Backing up the Hue Database

If you use the default SQLite database, then 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.

Configuring Hue to Access Another Database

Although SQLite is the default database type, some advanced users may prefer to have Hue access an alternate database type. Note that if you elect to configure Hue to use an external database, upgrades may require more manual steps in the future.

The following instructions are for MySQL, though you can also configure Hue to work with other common databases such as PostgreSQL and Oracle.

> **Tested Database Backends**
>
> Note that Hue has only been tested with SQLite and MySQL database backends.

Configuring Hue to Store Data in MySQL

> **Note**
>
> To use MySQL with Hue, you will need the package that allows you to compile other MySQL clients: the `mysql-devel` RPM or the `libmysqlclient-dev` Debian package.

To configure Hue to store data in MySQL:

1. Create a new database in MySQL and grant privileges to a Hue user to manage this 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)
```

2. Shut down Hue if it is running.
   To migrate your existing data to MySQL, use the following command to dump the existing database data to a text file. Note that using the ".json" extension is required.

```
$ /usr/share/hue/build/env/bin/hue dumpdata > <some-
temporary-file>.json
```

3. Open the `/etc/hue/hue.ini` file in a text editor.

   > **Note**
   >
   > Only the root user can edit this file.

   Directly below the [`database`] line, add the following options (and modify accordingly for your MySQL setup):

```
host=localhost
port=3306
engine=mysql
user=hue
password=secretpassword
name=hue
```

4. As the Hue user, configure Hue to load the existing data and create the necessary database tables:

```
$ /usr/share/hue/build/env/bin/hue syncdb --noinput
$ mysql -uhue -psecretpassword -e "DELETE FROM
hue.django_content_type;"
$ /usr/share/hue/build/env/bin/hue loaddata
<temporary-file-containing-dumped-data>.json
```

Your system is now configured and you can start the Hue server as normal.

Installing and Configuring Hue Shell

Hue includes the Shell application, which provides access to the Pig, Flume, and HBase command-line shells by default. The Shell application is designed to have the same look and feel as a Unix terminal. In addition to the shells configured by default, it is possible to include almost any process that exposes a command-line interface as an option in this Hue application.

> **Important**
>
> Pig Shell will not run unless you configure the `JAVA_HOME` environment variable in the `apps/shell/conf/hue-shell.ini` file. For instructions, see the following section, Modifying the Hue Shell Configuration File.

Verifying Hue Shell Installation

To work properly, Hue Shell requires the command-line shells to be installed and available on the system. You must specify absolute paths, so you should test exactly the commands you will enter in the configuration file in a terminal:

These are examples, you may have the binaries in different locations

To verify Pig Shell (Grunt) is installed:

```
$ /usr/bin/pig
```

To verify Flume Shell is installed:

```
$ /usr/bin/flume shell
```

To verify HBase Shell is installed:

```
$ /usr/bin/hbase shell
```

Modifying the Hue Shell Configuration File

To add or remove shells, modify the configuration for the application. The Hue Shell configuration is loaded from `apps/shell/conf/hue-shell.ini`, which uses the following format:

Format	Description
`[ shell ]`	This specifies the beginning of the Hue Shell configuration.

Format	Description
`shell_buffer_amount`	Optional. Amount of output to buffer for each shell in bytes. Defaults to 524288 (512 KiB) if not specified.
`shell_timeout`	Optional. Amount of time to keep shell subprocesses open when no open browsers refer to them. Defaults to 600 seconds (10 mins) if not specified.
`shell_write_buffer_limit`	Optional. Amount of pending commands to buffer for each shell, in bytes. Defaults to 10000 (10 KB) if not specified.
`shell_os_read_amount`	Optional. Number of bytes to specify to the read system call when reading subprocess output. Defaults to 40960 (usually 10 pages, since pages are usually 4096 bytes) if not specified.
`shell_delegation_token_dir`	Optional. If this instance of Hue is running with a Hadoop cluster with Kerberos security enabled, it must acquire the appropriate delegation tokens to execute subprocesses securely. The value under this key specifies the directory in which these delegation tokens are to be stored. Defaults to `/tmp/hue_shell_delegation_tokens` if not specified.
`[[ shelltypes ]]`	This sub-section groups the individual shell configurations.
`[[[ pig ]]]`	This section title is a key name that also begins the configuration parameters for a specific shell type ("pig" in this example). You can use any name, but it must be unique for each shell specified in the configuration file. Each key name denotes

Format	Description
	the beginning of a shell configuration section; each section can contain the following six parameters described in this table: `nice_name`, `command`, `help`, `environment`, and the value for the environment variable.
`nice_name = "Pig Shell (Grunt)"`	The user-facing name.
`command = "/usr/bin/pig -l /dev/null"`	The command to run to start the specified shell. The path to the binary must be an absolute path.
`help = "A platform for data exploration."`	Optional. A string that describes the shell.
`[[[[ environment ]]]]`	Optional. A section to specify environment variables to be set for subprocesses of this shell type. Each environment variable is itself another sub-section, as described below.
`[[[[[ JAVA_HOME ]]]]]`	The name of the environment variable to set. For example, Pig requires JAVA_HOME to be set.
`value = /usr/lib/jvm/java-6-sun`	The value for the environment variable.

Restrictions

While almost any process that exports a command-line interface can be included in the Shell application, processes that redraw the window, such as vim or top, cannot be exposed in this way.

Adding Shell types to Hue in the Cloudera Management Suite

If you are running Hue as a part of Cloudera Management Suite, you need to also modify the PERMISSION_ACTIONS tuple in `apps/shell/src/shell/settings.py`. For

each unique shell key from the `hue-shell.ini` config file, you must have an entry in the `PERMISSION_ACTIONS` tuple. Entries in this tuple are also tuples, where the first item of the tuple is the string "`launch_`" concatenated with the key, and the second item of the tuple is a string documenting the action. For example, the entry for pig is:

```
("launch_pig", "Launch the Pig Shell")
```

After you make the necessary changes, you must run

```
$ hue syncdb
```

to make your changes take effect and then restart Hue.

Unix User Accounts

To properly isolate subprocesses so as to guarantee security, you need a Unix user account for each Hue user who is using the Shell subprocess. The link between Hue users and Unix user accounts is the username, and so every Hue user who wants to use the Shell application must have a Unix user account with the same name on the server that runs Hue.

Also, there is a binary called `setuid` which provides a binary wrapper, allowing for the subprocess to be run as the appropriate user. In order to work properly for all users of Hue, this binary must be owned by root and must have the `setuid` bit set.

To make sure that these two requirements are satisfied, navigate to the directory with the `setuid` binary (`apps/shell/src/shell/`) and execute the following commands in a terminal:

Using sudo:

```
$ sudo chown root:hue setuid
$ sudo chmod 4750 setuid
$ exit
```

Using root:

```
$ su
# chown root:hue setuid
# chmod 4750 setuid
```

```
# exit
```

> **Important**
>
> If you are running Hue Shell against a secure cluster, see the Running Hue Shell against a Secure Cluster section for security configuration information for Hue Shell.

Running the Appropriate Web Server

Older versions of Hue shipped with the CherryPy web server as the default web server. This is no longer the case starting with CDH3 Update 1. In order to configure the default web server, you must modify `hue.ini` in `desktop/conf` and modify the value for `use_cherrypy_server`. This value must either be set to `false` or not specified in order for the Shell application to work.

Viewing the Hue and Hue Shell Documentation

For additional documentation about Hue, see the Hue manual and Hue SDK.

For additional documentation about using Hue Shell, see Hue Shell or the online Help in Hue for the Shell application.

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` respository before using the following commands to install or upgrade Pig. For instructions, see CDH3 Installation.

Upgrading Pig to CDH3 Update 2

The instructions that follow assume that you are upgrading Pig as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

To upgrade Pig to CDH 3 Update 2:

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.

Installing Pig

To install Pig on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-pig
```

To install Pig on Red Hat systems:

```
$ sudo yum install hadoop-pig
```

To install Pig on SUSE systems:

```
$ sudo zypper install hadoop-pig
```

> Pig automatically uses the active Hadoop configuration (whether standalone or pseudo-mode or other). After installing the Pig package, you can start the grunt shell.

To start the Grunt Shell:

```
$ pig
0   [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionE
ngine - Connecting to Hadoop file system at:
hdfs://localhost:8020
352 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionE
ngine - Connecting to map-reduce job tracker at:
localhost:9001
grunt>
```

To verify that the input and output directories from your example grep job exist (see CDH3 Deployment in Standalone Mode), list an HDFS directory from the Grunt Shell:

```
grunt> ls
hdfs://localhost/user/matt/input <dir>
hdfs://localhost/user/matt/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;
```

To check the status of your job while it is running, look at the JobTracker web console http://localhost:50030/.

Problem after Upgrading from CDH3 Update 0 to CDH3 Update 2

After you upgrade Pig from CDH3u0 to CDH3u2, the symbolic link `/etc/pig/conf` still points to `/etc/alternatives/pig`. It should point to `/etc/alternatives/pig-conf`. To fix this, follow instructions here.

Incompatible Changes as of the Pig 0.7.0 Release

Pig 0.7.0 contained several changes that are not backwards compatible with versions prior to 0.7.0; if you have scripts from a version of Pig prior to 0.7.0, you may need to modify your user-defined functions (UDFs) in order to work with the current Pig release. In particular, the Load and Store functions were changed. For information about updating your UDFs, see LoadStoreMigrationGuide and Pig070LoadStoreHowTo. For a list of all backward incompatible changes, see this page.

Using Pig with HBase

To allow Pig scripts to use HBase, add the following statement to the top of each script:

```
register /usr/lib/hbase/lib/zookeeper-3.3.3-cdh3u1.jar;
register /usr/lib/hbase/hbase-0.90.3-cdh3u1.jar;
```

```
register /usr/lib/hbase/lib/guava-r06.jar;
```

Viewing the Pig Documentation

For additional Pig documentation, see http://archive.cloudera.com/cdh/3/pig/.

Oozie Installation

Oozie is a server-based workflow engine specialized in running workflow jobs with actions that execute Hadoop jobs, such as MapReduce, Pig, Hive, Sqoop, HDFS operations, and sub-workflows. Oozie supports coordinator jobs, which is a sequence of workflow jobs that are created at a given frequency and start when all of the required input data is available. A command line client is also available that allows remote administration and management of Oozie jobs.

Upgrading Oozie to CDH3 Update 2

Note

- The following CDH3 versions of Oozie can be upgraded directly to CDH3 Update 2: Beta 4, Update 0, and Update 1.
- The following CDH3 versions of Hadoop work with the CDH3 Update 2 version of Oozie: Update 0, Update 1, and Update 2.

The instructions that follow assume that you are upgrading Oozie as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

To upgrade Oozie to CDH 3 Update 2, proceed as follows.

Step 1: Back Up Configuration Files and Database

Back up the Oozie configuration files (in `/etc/oozie` in an RPM or Debian installation) and the Oozie database.

Make sure you have a good copy of your current `oozie-site.xml`. For convenience you may want to save this copy in your home directory; you will need to edit it after installing the new version of Oozie (see Step 6).

Step 2: Stop the Oozie Server

To stop the Oozie Server:

- If the previous version of Oozie was installed from an RPM or Debian package:

```
sudo /sbin/service oozie stop
```

- If the previous version of Oozie was installed from a tarball:

```
sudo -u oozie /usr/lib/oozie/bin/oozie-stop.sh
```

Step 3: Install Oozie

Follow the procedure under Installing Oozie Tarball or Installing Oozie RPM or Debian Packages. For packaging information, see Oozie Packaging.

Step 4: Move Directories (tarball only)

If you are installing from a tarball:

- If you have used `~bin/oozie/oozie-env.sh` to configure Oozie to look for its directories (such as `conf`, `data`, `logs`, `run`, `tmp`, and so on) outside the Oozie expanded directory, copy your old `~bin/oozie/oozie-env.sh` file to the new Oozie expanded directory.
- If the Oozie directories are in their default locations in the Oozie expanded directory, copy or move these Oozie directories from the old Oozie expanded directory to the new Oozie expanded directory.

For more information, see Configuring Oozie.

Step 5: Run the Setup Script

To run the setup script:

```
sudo -u oozie /usr/lib/oozie/bin/oozie-setup.sh
```

> **Important**
>
> You *must* run `oozie-setup.sh`; otherwise the previous version of Oozie will continue to run.

You may also need to run the setup script to install the ExtJS library, and JDBC drivers for MySQL or Oracle. See Database Configuration and Enabling the Oozie Web Console.

Step 6: Edit the Configuration File

Edit the `oozie-site.xml` configuration file (the one you backed up in Step 1) so the the following two properties contain the following values. Copy the edited oozie-site.xml file to Oozie configuration directory (`/etc/oozie` in an RPM or Debian installation) .

```
    <property>

<name>oozie.service.ActionService.executor.ext.classes</n
ame>
        <value>

org.apache.oozie.action.hadoop.HiveActionExecutor,

org.apache.oozie.action.hadoop.SqoopActionExecutor,

org.apache.oozie.action.email.EmailActionExecutor,

org.apache.oozie.action.hadoop.DistcpActionExecutor
        </value>
    </property>

    <property>

<name>oozie.service.SchemaService.wf.ext.schemas</name>
        <value>hive-action-0.2.xsd,sqoop-action-
0.2.xsd,email-action-0.1.xsd,distcp-action-
0.1.xsd</value>
    </property>
```

Copy the edited `oozie-site.xml` file to the Oozie configuration directory.

Step 7: Start Oozie

See Starting, Stopping, and Accessing the Oozie Server.

The upgrade is now complete. For more information, see Configuring Oozie and Viewing the Oozie Documentation.

Oozie Packaging

There are three packaging options for installing Oozie:

- Tarball (`.tgz`) that contains both the Oozie server and the Oozie client.
- Separate RPM packages for Oozie server (`oozie`) and client (`oozie-client`)
- Separate Debian packages for Oozie server (`oozie`) and client (`oozie-client`)

Oozie Prerequisites

- Prerequisites for installing Oozie server:
  - A Unix-like system (tested on Centos 5.5, Ubuntu 10.04, SUSE Linux Enterprise Server 11, OS X 10.6)
  - Java 1.6+ (tested with JRE 1.6.0_20)
- Prerequisites for installing Oozie client:
  - Java 1.6+ (tested with JRE 1.6.0_20)

Installing Oozie Tarball

The Oozie tarball contains both Oozie server and Oozie client. You can download the Oozie tarball from this page.

To install the Oozie tarball:

1. Create a Unix group called `oozie` and a Unix user called `oozie` with the group `oozie` as its primary group.
2. Unpack the tarball in the appropriate directory. For example:

```
$ (cd /usr/lib/ && sudo tar -zxvf
<PATH_TO_OOZIE_TAR_GZ>)
```

3. Change ownership of the Oozie installation to `oozie:oozie`:

```
$ sudo chown -R oozie:oozie /usr/lib/oozie-2.3.0+14-1
```

4. Create a softlink without the version:

```
$ sudo ln -s /usr/lib/oozie-2.3.0+14-1 oozie
```

5. Soft-link the `/usr/lib/oozie/bin/oozie` command line tool in the `/usr/bin` directory:

```
$ cd /usr/lib/bin
$ sudo ln -s /usr/lib/oozie/bin/oozie oozie
```

   Once laid out on the file system, the resulting directory will contain all the necessary client and server files.

> **Note**
>
> To install only the Oozie client, follow the above procedure with the {{oozie-client.tar.g
> z}} tarball bundled in the Oozie tarball, but you do not need to create the `oozie` Unix user and group.

You are now ready to configure Oozie. For instructions, see Configuring Oozie.

Installing Oozie RPM or Debian Packages

Oozie is distributed as two separate packages; a client package (`oozie-client`) and and a server package (`oozie`). Depending on what you are planning to install, choose the appropriate package and install it with your preferred package manager application.

> **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 CDH3 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 system:

```
$ sudo yum install oozie
```

To install the Oozie client package on a Red Hat system:

```
$ sudo yum install oozie-client
```

To install the Oozie server package on a SUSE system:

```
$ sudo zypper install oozie
```

To install the Oozie client package on a SUSE system:

```
$ sudo zypper install oozie-client
```

> **Note**
>
> The RPM and Debian `oozie` package installation creates an `oozie` service configured to start Oozie at system start up time.

You are now ready to configure Oozie. See the next section.

Configuring Oozie

The location of Oozie configuration, documentation, examples, and runtime files depends on how Oozie is installed as shown in the following table. When you install

Oozie from the tarball, the Oozie server creates all configuration, documentation, and runtime files in directories in the installation directory. When you install Oozie from a RPM or Debian package, the Oozie server creates all configuration, documentation, and runtime files in the standard Unix directories.

Files	**Tarball Installation**	**RPM/Debian Installation**
binaries	`<EXPANDED_DIR>/`	`/usr/lib/oozie/`
configuration	`<EXPANDED_DIR>/`	`/etc/oozie/`
documentation	`<EXPANDED_DIR>/`	for SLES: `/usr/share/doc/packages/oozie/` for other platforms: `/usr/share/doc/oozie/`
examples TAR.GZ	`<EXPANDED_DIR>/`	for SLES: `/usr/share/doc/packages/oozie/` for other platforms: `/etc/oozie/doc/oozie/`
sharelib TAR.GZ	`<EXPANDED_DIR>/`	`/usr/lib/oozie/`
data	`<EXPANDED_DIR>/data` and `<EXPANDED_DIR>/oozie-server/`	`/var/lib/oozie/`
logs	`<EXPANDED_DIR>/log`	`/var/log/oozie/`
temp	`<EXPANDED_DIR>/oozie-server/temp/`	`/var/tmp/oozie/`
PID file	`<EXPANDED_DIR>/oozie-server/temp/`	`/var/run/oozie/`

To customize the location of these files when installing Oozie from a tarball, copy the `<EXPANDED_DIR>/src/cloudera/oozie-env.sh` file in the `<EXPANDED_DIR>/bin` directory, edit the copied file, and set the environment variables to the desired values. Make sure the specified directories exist and that the Unix user `oozie` owns them.

Hadoop Configuration

CDH 3 is preconfigured to work with the Cloudera Oozie distribution if you install Oozie following the instructions on this page.

Database Configuration

Oozie is preconfigured to use Apache Derby as its database, but you can also configure Oozie to use Postgres, MySQL, or Oracle.

Configuring Oozie to Use Postgres

Use the procedure that follows to configure Oozie to use Postgres instead of Apache Derby.

Step 1: Install Postgres 8.4.x or 9.0.x.

Step 2: Create the Oozie user and Oozie database.

For example, using the Postgres `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.UTF-8''
 LC_CTYPE = 'en_US.UTF-8''
 CONNECTION LIMIT = -1;
CREATE DATABASE

postgres=# \q
```

Step 3: Configure Postgres to accept network connections for user oozie.

Edit the Postgres `data/pg_hba.conf` file as follows:

```
host    oozie         oozie         0.0.0.0/0
md5
```

Step 4: Reload the Postgres configuration.

```
$ sudo -u postgres pg_ctl reload -s -D
/opt/PostgresSQL/8.4/data
```

Step 5: Configure Oozie to use Postgres.

Edit the `oozie-site.xml` file as follows:

```
    ...
    <property>

<name>oozie.service.StoreService.jdbc.driver</name>
        <value>org.postgresql.Driver</value>
    </property>
    <property>
        <name>oozie.service.StoreService.jdbc.url</name>

<value>jdbc:postgresql://localhost:5432/oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

> **Note**
>
> In the JDBC URL property, replace `localhost` with the hostname where Postgres is running.
>
> In the case of Postgres, 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.

Step 6: Restart Oozie.

Restart Oozie and check the logs to make sure Oozie has started successfully using the Postgres database.

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 5.0.x (Currently MySQL 5.5.x is not supported).

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.StoreService.jdbc.driver</name>
        <value>com.mysql.jdbc.Driver</value>
    </property>
    <property>
        <name>oozie.service.StoreService.jdbc.url</name>
        <value>jdbc:mysql://localhost:3306/oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.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.

Use the `oozie-setup.sh` command-line tool; for example:

```
$ oozie-setup.sh -jar mysql-connector-java-5.1.6.jar -
```

```
extjs /tmp/ext-2.2.zip
```

> **Note**
>
> The MySQL JDBC driver and the ExtJS library must be manually downloaded. The ExtJS library is needed by the web console; see Enabling the Oozie Web Console.

Step 5: Restart Oozie.

Restart Oozie and check the logs to make sure Oozie has started successfully using the MySQL database.

Configuring Oozie to use Oracle

Use the procedure that follows to configure Oozie to use Oracle 11g instead of Apache Derby.

Step 1: Install and start Oracle 11g.

Step 2: Create the Oozie Oracle user.

For example, using 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.StoreService.jdbc.driver</name>
        <value>oracle.jdbc.driver.OracleDriver</value>
    </property>
    <property>
        <name>oozie.service.StoreService.jdbc.url</name>

<value>jdbc:oracle:thin://localhost:1521/oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.jdbc.username</name>
        <value>oozie</value>
    </property>
    <property>

<name>oozie.service.StoreService.jdbc.password</name>
        <value>oozie</value>
    </property>
    ...
```

> **Note**
>
> In the JDBC URL property, replace `localhost` 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.

Use the `oozie-setup.sh` command-line tool; for example:

```
$ oozie-setup.sh -jar ojdbc6.jar -extjs /tmp/ext-2.2.zip
```

> **Note**
>
> The Oracle JDBC driver and the ExtJS library must be manually downloaded. The ExtJS library is needed by the web console; see Enabling the Oozie Web Console.

Step 5: Restart Oozie.

Restart Oozie and check the logs to ensure Oozie has started successfully using the Oracle 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.

> **Important**
>
> You must specify all the items you need to configure each time you run `oozie-setup.sh`. If you have already run `oozie-setup.sh` with items you need for the current installation, and you now need to add the ExtJS library, you must specify all the items you specified before, in addition to ExtJS. *Do not* re-run `oozie-setup.sh` with just the `-extjs` argument; otherwise you will unconfigure any other items you specified on the previous run.
>
> For example, if you have already run a command such as the following to configure the Oracle JDBC driver,
>
> ```
> $ oozie-setup.sh -jar ojdbc6.jar
> ```
>
> run a command like this to add the ExtJS library:
>
> ```
> $ oozie-setup.sh -jar ojdbc6.jar -extjs /tmp/ext-2.2.zip
> ```

1. Download the ExtJS version 2.2 library from http://extjs.com/deploy/ext-2.2.zip and place it a convenient location.
2. Add the ExtJS library to Oozie using the `oozie-setup.sh` script; for example (if ExtJS is the only item you need to configure):

```
$ sudo -u oozie /usr/lib/oozie/bin/oozie-setup.sh -
extjs /tmp/ext-2.2.zip
```

Configuring Oozie with Kerberos Security

To configure Oozie with Kerberos security, see Oozie Security Configuration.

Installing Oozie ShareLib in Hadoop HDFS

The Oozie installation bundles Oozie ShareLib, which contains all of the necessary JARs to enable workflow jobs to run Streaming/Pig/Hive/Sqoop actions.

> **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 `hadoop fs -put` command in the following instructions, you must run the `sudo -u oozie kinit -k -t /etc/oozie/oozie.keytab` command.

To install Oozie ShareLib in Hadoop HDFS in the `oozie` user home directory:

```
$ mkdir /tmp/ooziesharelib
$ cd /tmp/ooziesharelib
$ tar xzf /usr/lib/oozie/oozie-sharelib.tar.gz
$ sudo -u oozie hadoop fs -put share /user/oozie/share
$ rm -rf /tmp/ooziesharelib
```

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 by using the following commands:

Tarball installation:

```
$ sudo -u oozie /usr/lib/oozie/bin/oozie-start.sh
```

RPM or Debian installation:

```
$ sudo /sbin/service oozie start
```

When you start the Oozie server for the first time, it automatically creates the necessary database schema.

If you see the message `Oozie System ID [oozie-tomc] started` in the `oozie.log` log file, the system started successfully.

> **Note**
>
> By default, Oozie server runs on port `11000` and its URL is `http://<OOZIE_HOSTNAME>:11000/oozie`.

Stopping the Oozie Server

Tarball installation:

```
$ sudo -u oozie /usr/lib/oozie/bin/oozie-stop.sh
```

RPM or Debian installation:

```
$ sudo /sbin/service oozie stop
```

Accessing the Oozie Server with the Oozie Client

The Oozie client is invoked using the `/usr/bin/oozie` script. The Oozie client is a command-line utility that interacts with the Oozie server via Oozie web-services API.

> **Important**
>
> If Oozie is configured with Kerberos Security enabled, you must have a Kerberos session. For example, by running the `kinit` command.

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 called `OOZIE_URL` pointing to the URL of the Oozie server. Then the `-oozie` option can be skipped.

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: 2.3.0-CDH3B4
```

Accessing the Oozie Server with a Browser

If you enabled the Oozie web console by adding the ExtJS library, the console is accessible at `http://<OOZIE_HOSTNAME>:11000/oozie`.

> **Note**
>
> If Oozie Server is configured to use Kerberos HTTP SPNEGO Authentication for its users, a web browser that supports Kerberos HTTP SPNEGO must be used (for example, Firefox or Internet Explorer).

Viewing the Oozie Documentation

For additional Oozie documentation, see http://archive.cloudera.com/cdh/3/oozie.

Hive Installation

Apache Hive is a powerful data warehousing application built on top of Hadoop which enables you to access your data using Hive QL, a language that is similar to SQL.

Install Hive on your client machine where you submit jobs; you don't need to install it on your servers.

> **Important**
>
> If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` respository before using the following commands to install Hive. For instructions, see CDH3 Installation.

Upgrading Hive in CDH3

If you used Hive in versions prior to CDH3 Beta 4, read this section. CDH3 includes changes in the Hive MetaStore schema that are part of the Hive 0.7 release. If you want to use Hive in CDH3, it is imperative that after you install Hive 0.7, you upgrade the Hive MetaStore schema by running the appropriate schema upgrade script located in the `/usr/lib/hive/metastore/scripts/upgrade` directory in the Hive installation. Scripts for Derby and MySQL databases are available. If you are using a different database for your MetaStore, you will need to provide your own upgrade script.

> **Important**
>
> The name of the default Derby database changed in CDH3 Production. If you used the default Derby database configuration in a previous release of CDH, you will need to change the name of the Derby database directory from `/usr/lib/hive/metastore/${user.name}_db` to `/usr/lib/hive/metastore/metastore_db`.

Also, the MySQL JDBC connector is no longer bundled with Hive. If you want to use MySQL as your MetaStore, you can either install the package that provides this (as provided by your distribution), or download the MySQL JDBC connector from http://www.mysql.com/downloads/connector/j/ and place the JAR file in the `/usr/lib/hive/lib` directory.

Upgrading Hive from CDH3 Update 1 to CDH3 Update 2

The instructions that follow assume that you are upgrading Hive as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

To upgrade Hive to CDH 3 Update 2, proceed as follows.

Step 1: Stop Hive Processes

Exit the Hive console and make sure no Hive scripts are running.

Step 2: Install the new Hive version and Restart the Console

See the next section, Installing Hive.

The upgrade is now complete.

Installing Hive

To install Hive on Ubuntu and other Debian systems:

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

To install Hive on Red Hat systems:

```
$ sudo yum install hadoop-hive
```

To install Hive on SUSE systems:

```
$ sudo zypper install hadoop-hive
```

To start the Hive console:

```
$ hive
hive>
```

Configuring the Hive Metastore

In order to make setup easy for new users, Hive's Metastore is configured to store metadata locally in an embedded Apache Derby database. Unfortunately, this configuration only allows a single user to access the Metastore at a time. Cloudera strongly encourages users to use a MySQL database instead. This section describes how to configure Hive to use a remote MySQL database, which allows Hive to support multiple users. See the Hive Metastore documentation for additional information.

Prerequisites

Install the MySQL JDBC Connector in the Hive lib directory:

```
$ curl -L 'http://www.mysql.com/get/Downloads/Connector-
J/mysql-connector-java-
5.1.15.tar.gz/from/http://mysql.he.net/' | tar xz
$ sudo cp mysql-connector-java-5.1.15/mysql-connector-
java-5.1.15-bin.jar /usr/lib/hive/lib/
```

The MySQL administrator should create the initial database schema using the `hive-schema-0.7.0.mysql.sql` file located in the `/usr/lib/hive/scripts/metastore/upgrade/mysql` directory.

```
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE
/usr/lib/hive/scripts/metastore/upgrade/mysql/hive-
schema-0.7.0.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:

```
mysql> CREATE USER 'hiveuser'@'%' IDENTIFIED BY
'password';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON metastore.*
TO 'hiveuser'@'%';
mysql> REVOKE ALTER,CREATE ON metastore.* FROM
'hiveuser'@'%';
```

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 your Metastore schema when they use older or newer versions of Hive.

Hive Configuration

You must add or modify the following configuration parameters in `/etc/hive/conf/hive-site.xml`. The `javax.jdo.option.ConnectionURL` parameter specifes the URL of the MySQL database. The `javax.jdo.option.ConnectionUserName` and `javax.jdo.option.ConnectionPassword` parameters should be set to the user name and password for the MySQL user. For example, given a MySQL database running on `MYHOST` and the user account `hiveuser` with the password `password`, the configuration would be set as follows:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://MYHOST/metastore</value>
</property>

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

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

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

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

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

Using Hive with HBase

To allow Hive scripts to use HBase, add the following statements to the top of each script:

```
add jar /usr/lib/hbase/lib/zookeeper-3.3.3-cdh3u1.jar
add jar /usr/lib/hbase/hbase-0.90.3-cdh3u1.jar;
add jar /usr/lib/hbase/lib/guava-r06.jar;
```

Viewing the Hive Documentation

For additional Hive documentation, see http://archive.cloudera.com/cdh/3/hive/.

> **For More Information**
>
> To view Cloudera's video tutorial about using Hive, see "Hadoop Training: Introduction to Hive".

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.

> **Important**
>
> If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` respository before using the following commands to install or upgrade HBase. For instructions, see CDH3 Installation.

Upgrading HBase to CDH3 Update 2

The instructions that follow assume that you are upgrading HBase as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

To upgrade Hbase to CDH 3 Update 2, proceed as follows:

Step 1: Stop the Thrift Server and Clients

To stop the Thrift server and clients:

```
sudo service hadoop-hbase-thrift stop
```

Step 2: Stop the Cluster

To stop the cluster:

Use the following command on the master node:

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

This shuts down the master and the region servers gracefully.

Step 3: Stop the ZooKeeper Server

To stop the ZooKeeper server:

```
$ sudo service hadoop-zookeeper-server stop
```

Note

Depending on your platform and release, you may need to use

```
$ sudo /sbin/service hadoop-zookeeper-server stop
```

or

```
$ sudo /sbin/service hadoop-zookeeper stop
```

> **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 to CDH3 Update 2

Step 4: Install the new version of HBase

Follow directions in the next section, Installing HBase

The upgrade is now complete.

Installing HBase

To install HBase on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-hbase
```

To install HBase on Red Hat systems:

```
$ sudo yum install hadoop-hbase
```

To install HBase on SUSE systems:

```
$ sudo zypper install hadoop-hbase
```

To list the installed files on Ubuntu and other Debian systems:

```
$ dpkg -L hadoop-hbase
```

To list the installed files on Red Hat and SUSE systems:

```
$ rpm -ql hadoop-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`).

```
HBase wrapper script              /usr/bin/hbase
HBase Configuration Files         /etc/hbase/conf
HBase Jar and Library Files       /usr/lib/hbase
HBase Log Files                   /var/log/hbase
HBase service scripts             /etc/init.d/hadoop-
hbase-*
```

You are now ready to enable the server daemons you want to use with Hadoop. Java-based client access is also available by adding the jars in `/usr/lib/hbase/` and `/usr/lib/hbase/lib/` to your Java class path.

Host Configuration Settings 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 machine 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 choose a different name server than the system-wide default.

Using the Network Time Protocol (NTP) with HBase

The clocks on cluster members should be in basic alignments. Some skew is tolerable, but excessive skew could generate odd behaviors. Run NTP on your cluster, or an equivalent. If you are having problems querying data or unusual cluster operations, verify the system time.

Setting User Limits for HBase

Because HBase is a database, it uses a lot of files at the same time. The default ulimit setting of 1024 for the maximum number of open files on Unix systems is insufficient. Any significant amount of loading will result in failures in strange ways and cause the error message `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 HBase FAQ. 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
```

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.

If you are using ulimit, you must make the following configuration changes:

1. In the `/etc/security/limits.conf` file, add the following lines:

   Note

   Only the root user can edit this file.

   ```
   hdfs  -       nofile  32768
   hbase  -       nofile  32768
   ```

2. To apply the changes in `/etc/security/limits.conf` on Ubuntu and other Debian systems, add the following line in the `/etc/pam.d/common-session` file:

   ```
   session required  pam_limits.so
   ```

Using dfs.datanode.max.xcievers 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 property is called `dfs.datanode.max.xcievers` (the property is spelled in the code exactly as shown here). Before loading, make sure you have configured the value for `dfs.datanode.max.xcievers` in the `conf/hdfs-site.xml` file to at least 4096 as shown below:

```
<property>
  <name>dfs.datanode.max.xcievers</name>
  <value>4096</value>
</property>
```

Be sure to restart HDFS after changing the value for `dfs.datanode.max.xcievers`. If you don't change that value as described, strange failures can occur and an error message about exceeding the number of `xcievers` will be added to the DataNode logs. Other error messages about missing blocks are also logged, such as:

```
10/12/08 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...
```

Starting HBase in Standalone 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. In order to run HBase in standalone mode, you must install the HBase Master package:

Installing the HBase Master for Standalone Operation

To install the HBase Master on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-hbase-master
```

To install the HBase Master on Red Hat systems:

```
$ sudo yum install hadoop-hbase-master
```

To install the HBase Master on SUSE systems:

```
$ sudo zypper install hadoop-hbase-master
```

Starting the HBase Master

On Red Hat and SUSE systems (using `.rpm` packages) you can start now start the HBase Master by using the included service script:

```
$ sudo /etc/init.d/hadoop-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 just 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 uninstall the hadoop-zookeeper-server package.

Accessing HBase by using the HBase Shell

After you have started the standalone installation, you can access the database by using the HBase Shell:

```
$ hbase shell
HBase Shell; enter 'help<RETURN>' for list of supported
commands.
Type "exit<RETURN>" to leave the HBase Shell
Version: 0.89.20100621+17, r, Mon Jun 28 10:13:32 PDT
2010

hbase(main):001:0> status 'detailed'
version 0.89.20100621+17
0 regionsInTransition
1 live servers
    my-machine:59719 1277750189913
        requests=0, regions=2, usedHeap=24, maxHeap=995
        .META.,,1
            stores=2, storefiles=0, storefileSizeMB=0,
memstoreSizeMB=0, storefileIndexSizeMB=0
        -ROOT-,,0
            stores=1, storefiles=1, storefileSizeMB=0,
memstoreSizeMB=0, storefileIndexSizeMB=0
0 dead servers
```

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` here.

When getting an `Configuration` object for a HBase MapReduce job, instantiate it using the `HBaseConfiguration.create()` method.

Configuring HBase in Pseudo-distributed Mode

Pseudo-distributed mode differs from *standalone* mode in that each of the component processes run in a separate JVM.

> **Note**
>
> If the HBase master is already running in standalone mode, stop it by running `/etc/init.d/hadoop-hbase-master stop` before continuing with pseudo-distributed configuration.

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. Be sure to replace `localhost` with the host name of your HDFS Name Node if it is not running locally.

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://localhost/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
```

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:

- ZooKeeper Quorum Peers
- HBase Master
- HBase Region Server

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 colocated 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 on a Single Server" section of *ZooKeeper Installation*.

Starting the HBase Master

After ZooKeeper is running, you can start the HBase master in standalone mode.

```
$ sudo /etc/init.d/hadoop-hbase-master start
```

Starting an HBase Region Server

The Region Server 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 Region Server on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-hbase-regionserver
```

To enable the HBase Region Server on Red Hat systems:

```
$ sudo yum install hadoop-hbase-regionserver
```

To enable the HBase Region Server on SUSE systems:

```
$ sudo zypper install hadoop-hbase-regionserver
```

To start the Region Server:

```
$ sudo /etc/init.d/hadoop-hbase-regionserver start
```

Verifying the Pseudo-Distributed Operation

After you have started ZooKeeper, the Master, and a Region Server, 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 Java JDK, which you can obtain by installing the Java JDK, available 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 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 colocated along with the region servers, but should not be colocated with the NameNode or the Job Tracker. For more information about Thrift, visit http://incubator.apache.org/thrift/.

To enable the HBase Thrift Server on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-hbase-thrift
```

To enable the HBase Thrift Server on Red Hat systems:

```
$ sudo yum install hadoop-hbase-thrift
```

To enable the HBase Thrift Server on SUSE systems:

```
$ sudo zypper install hadoop-hbase-thrift
```

Deploying HBase in a Distributed Cluster

After you have HBase running in pseudo-distributed mode, the same configuration can be extended to running on a distributed cluster.

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 and a DataNode. When collocating with Task Trackers, 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 may 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. 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>
```

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 Region Servers

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.

Troubleshooting

The Cloudera packages of HBase have been configured to place logs in `/var/log/hbase`. While getting started, Cloudera recommends tailing these logs to note any error messages or failures.

Viewing the HBase Documentation

For additional HBase documentation, see http://archive.cloudera.com/cdh/3/hbase/.

ZooKeeper Installation

Apache ZooKeeper is a highly reliable and available service that provides coordination between distributed processes.

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/.

Upgrading ZooKeeper to CDH3 Update 2

Cloudera recommends that you use a **rolling upgrade** process to upgrade ZooKeeper: that is, upgrade one server in the ZooKeeper cluster 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 cluster 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 an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

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 /sbin/service hadoop-zookeeper-server stop
```

or

```
$ sudo /sbin/service hadoop-zookeeper stop
```

depending on the platform and release.

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.

> **Note**
>
> Do not try to start the server yet.

Step 4: Re-enable the Server

Because of a packaging problem in earlier releases, you need to re-enable the server manually after upgrading ZooKeeper to CDH3 Update 2:

```
$ sudo /sbin/chkconfig --add hadoop-zookeeper-server
```

Step 5: 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 6: Upgrade the Remaining Nodes

Repeat Steps 1-5 above on each of the remaining nodes.

The ZooKeeper cluster upgrade is now complete.

Installing the ZooKeeper Packages

There are two ZooKeeper server packages:

- The `hadoop-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 `hadoop-zookeeper-server` package contains the `init.d` scripts necessary to run ZooKeeper as a daemon process. Because `hadoop-zookeeper-server` depends on `hadoop-zookeeper`, installing the server package automatically installs the base package.

> **Important**
>
> If you have not already done so, install Cloudera's `yum`, `zypper/YaST` or `apt` respository before using the following commands to install ZooKeeper. For instructions, see CDH3 Installation.

Installing the ZooKeeper Base Package

To install ZooKeeper on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-zookeeper
```

To install ZooKeeper on Red Hat systems:

```
$ sudo yum install hadoop-zookeeper
```

To install ZooKeeper on SUSE systems:

```
$ sudo zypper install hadoop-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 for more information.

To install a ZooKeeper server on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-zookeeper-server
```

To install ZooKeeper on Red Hat systems:

```
$ sudo yum install hadoop-zookeeper-server
```

To install ZooKeeper on SUSE systems:

```
$ sudo zypper install hadoop-zookeeper-server
```

To start ZooKeeper

> **Note**
>
> ZooKeeper may start automatically on installation on Ubuntu and other Debian systems.

Use the following command to start ZooKeeper:

```
$ sudo /sbin/service hadoop-zookeeper-server start
```

Installing ZooKeeper in a Production Environment

For use in a production environment, you should deploy ZooKeeper as a cluster with an odd number of nodes. As long as a majority of the servers in the cluster are available, the ZooKeeper service will be available. The minimum recommended cluster size is three ZooKeeper servers, and it is recommended that each server run on a separate machine.

ZooKeeper deployment on multiple servers requires a bit of additional configuration. The configuration file (`zoo.cfg`) on each server must include a list of all servers in the

cluster, and each server must also have a `myid` file in its data directory (by default `/var/zookeeper`) that identifies it as one of the servers in the cluster.

For instructions describing how to set up a multi-server deployment, see Installing a Multi-Server Setup.

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.

The ZooKeeper server does not remove the snapshots and log files, so they will accumulate over time. You will need to cleanup this directory occasionally, based on your backup schedules and processes. To automate the cleanup, a `zkCleanup.sh` script is provided in the `bin` directory of the `hadoop-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/cdh/3/zookeeper/.

Whirr Installation

Apache Whirr is a set of libraries for running cloud services. You can use Whirr to run CDH3 clusters on cloud providers' clusters, such as Amazon Elastic Compute Cloud (Amazon EC2). There's no need to install the RPMs for CDH3 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` respository before using the following commands to install or update Whirr. For instructions, see CDH3 Installation.

Upgrading Whirr to CDH3 Update 2

The instructions that follow assume that you are upgrading Whirr as part of an upgrade to CDH3 Upgrade 2, and have already performed the steps under Upgrading to CDH3.

> **Important**
>
> The upgrade from CDH3 Update 1 to CDH3 Update 2 replaces Whirr version 0.3.0 with version 0.5.0 As a result, the properties used in the configuration file (called `hadoop.properties` in these instructions) have changed, and you need to make changes as described in Step 2 below.

To upgrade Whirr to CDH 3 Update 2, proceed as follows.

Step 1: Stop the Whirr proxy

To stop the Whirr proxy

Kill the `hadoop-proxy.sh` process by pressing Control-C.

Step 2: Update the Properties File

Edit the configuration file, called called `hadoop.properties` in these instructions, and save it.

- For Hadoop, change the following lines:

```
whirr.hadoop-install-runurl=cloudera/cdh/install
whirr.hadoop-configure-runurl=cloudera/cdh/post-configure
```

to

```
whirr.hadoop-install-function=install_cdh_hadoop
whirr.hadoop-configure-function=configure_cdh_hadoop
```

- For HBase, change:

```
whirr.hbase.install-function=install_cdh_hbase
whirr.hbase.configure-function=configure_cdh_hbase
```

- For ZooKeeper, change:

```
whirr.zookeeper-install-function=install_cdh_zookeeper
whirr.zookeeper-configure-
function=configure_cdh_zookeeper
```

See Defining a Whirr Cluster for a sample file.

Step 3: Destroy the Cluster

Whirr clusters are normally short-lived. If you have a running cluster, destroy it: see Destroying a cluster.

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.

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 SUSE 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

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.

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=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.hadoop-install-function=install_cdh_hadoop
whirr.hadoop-configure-function=configure_cdh_hadoop
```

> **Note**
>
> For information explaining how to find your cloud credentials, see the Whirr FAQ.

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-0.20/conf.empty /etc/hadoop-
0.20/conf.whirr
$ rm -f /etc/hadoop-0.20/conf.whirr/*-site.xml
$ cp ~/.whirr/myhadoopcluster/hadoop-site.xml
/etc/hadoop-0.20/conf.whirr
```

2. If you are using a Ubuntu (or other Debian) or SUSE system, type these commands:

```
$ sudo update-alternatives --install /etc/hadoop-
0.20/conf hadoop-0.20-conf /etc/hadoop-
0.20/conf.whirr 50
$ update-alternatives --display hadoop-0.20-conf
```

3. If you are using a Red Hat system, type these commands:

```
$ sudo alternatives --install /etc/hadoop-0.20/conf
hadoop-0.20-conf /etc/hadoop-0.20/conf.whirr 50
$ alternatives --display hadoop-0.20-conf
```

4. You can now browse HDFS:

```
$ hadoop fs -ls /
```

To run a MapReduce job, run these commands:

```
$ export HADOOP_HOME=/usr/lib/hadoop
$ hadoop fs -mkdir input
$ hadoop fs -put $HADOOP_HOME/CHANGES.txt input
$ hadoop jar $HADOOP_HOME/hadoop-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.

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.

Upgrading Snappy to CDH3 Update 2

To upgrade Snappy, simply install the new `hadoop-0.20-native` package if you haven't already done so; for instructions, see the next section, Snappy Installation.

Snappy Installation

Snappy is provided in the native package along with the other native libraries (such as native `gzip` compression). If you are already using this package there is no additional installation to do. Otherwise follow these installation instructions:

To install Snappy on Ubuntu and other Debian systems:

```
$ sudo apt-get install hadoop-0.20-native
```

To install Snappy on Red Hat systems:

```
$ sudo yum install hadoop-0.20-native
```

To install Snappy on SUSE systems:

```
$ sudo zypper install hadoop-0.20-native
```

> **Note**
>
> If you install Hadoop from a tarball, only 64-bit Snappy libraries are available. If you need to use Snappy on 32-bit platforms, use the other packages, such as Red Hat or Debian packages.

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`:

```
<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>
```

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.

Property	Description
`mapred.output.compress`	Whether to compress the final job outputs (`true` or `false`)
`mapred.output.compression.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`	For `SequenceFile` outputs, what type of compression should be used (`NONE`, `RECORD`, or `BLOCK`). `BLOCK` is recommended.

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.comp
ress.SnappyCodec;
SET mapred.output.compression.type=BLOCK;
```

Configuring Flume to use Snappy Compression

Depending on the architecture of the machine you are installing on, add one of the following lines to `/usr/lib/flume/bin/flume-env.sh`:

- For 32-bit platforms:

```
export
JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-i386-
32
```

- For 64-bit platforms:

```
export
JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-
64
```

The following section explains how to take advantage of Snappy compression.

Using Snappy compression in Flume Sinks

You can specify Snappy as a compression codec in Flume's configuration language. For example, the following specifies a Snappy-compressed `SequenceFile` sink on HDFS:

```
customdfs("hdfs://namenode/path", seqfile("snappy"))
```

> **Warning**
>
> When using Snappy compression with `SequenceFile` or Avro data files in Flume you should **not** set the `flume.collector.dfs.compress.codec` property, otherwise the files will be "double compressed"; that is, the blocks inside the file and the file container itself will be compressed. Some components (e.g. Pig, Hive) may have difficulty reading such files.

Using Snappy compression in Sqoop Imports

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.

Configuring HBase to use Snappy Compression

> **Important**
>
> You need to configure HBase to use Snappy only if you installed Hadoop and HBase from tarballs; if you installed them from RPM or Debian packages, Snappy requires no HBase configuration.

Depending on the architecture of the machine you are installing on, add one of the following lines to `/etc/hbase/conf/hbase-env.sh`:

- For 32-bit platforms:

```
export
HBASE_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-i386-
32
```

- For 64-bit platforms:

```
export
HBASE_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-
amd64-64
```

To use Snappy compression in HBase Tables, specify the column family compression as `snappy`. For example, in the shell:

```
create 'mytable', {NAME=>'mycolumnfamily:',
COMPRESSION=>'snappy'}
```

Viewing the Snappy Documentation

For more information about Snappy, see http://code.google.com/p/snappy/.

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.

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

Mahout Prerequisites

- A Unix-like system (tested on Centos 5.5+, SLES11 and Ubuntu 9.04+)
- Java 1.6.x (tested only with JDK 1.6. See Java Development Kit Installation.)
- CDH3 Update 2

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 CDH3 Installation.

Mahout Installation

You can install Mahout from an RPM or Debian package, or from a tarball. 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.

To install Mahout on a Red Hat system:

```
$ sudo yum install mahout
```

To install Mahout on an Ubuntu or other Debian system:

```
$ sudo apt-get install mahout
```

To install Mahout on a SUSE system:

```
$ sudo zypper install mahout
```

To install Mahout on a system for which packages are not available:

Download a Mahout tarball from here.

Mahout Documentation

For more information about Mahout, see the Apache Mahout wiki.

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 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).

Starting with CDH3 Update 2, all components in CDH3 that produce or consume files support Avro data files as a file format. Bear in mind, however, since 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 CDH3 components.

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. If you choose to use Snappy compression you need to install Snappy first.

You do not need to do any additional configuration to read a compressed Avro data file rather 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

You can specify Avro data files as an output format in Flume's configuration language. For example, the following specifies an Avro data file sink on HDFS:

```
customdfs("hdfs://namenode/path", avrodata())
```

The Avro data file's schema has fields for the timestamp, priority, hostname, and fields, and the body of the message is a `"bytes"` field.

To enable Snappy compression, use:

```
customdfs("hdfs://namenode/path", avrodata("snappy"))
```

Sqoop

On the command line, use the following option to import to Avro data files:

```
--as-avrodatafile
```

Sqoop 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
org.apache.hadoop.io.compress.SnappyCodec
```

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>1.5.4</version>
</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 contrib/piggybank/java/piggybank.jar
REGISTER contrib/piggybank/java/lib/avro-1.5.4.jar
REGISTER contrib/piggybank/java/lib/jackson-core-asl-
1.7.3.jar
REGISTER contrib/piggybank/java/lib/jackson-mapper-asl-
1.7.3.jar
REGISTER contrib/piggybank/java/lib/json-simple-1.1.jar
REGISTER contrib/piggybank/java/lib/snappy-java-
1.0.3.2.jar
```

Then you can load Avro data files as follows:

```
a = LOAD 'my_file.avro' USING
org.apache.pig.piggybank.storage.avro.AvroStorage();
```

Pig will map 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 will generate 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 enable Snappy compression on output files do the following before issuing the `STORE` statement:

```
SET mapred.output.compress true
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

This table definition demonstrates how to create a Hive table that is backed by Avro data files:

```
CREATE TABLE my_avro_table(notused INT)
  ROW FORMAT SERDE
  'com.linkedin.haivvreo.AvroSerDe'
  WITH SERDEPROPERTIES (
    'schema.url'='file:///tmp/schema.avsc')
  STORED as INPUTFORMAT
  'com.linkedin.haivvreo.AvroContainerInputFormat'
  OUTPUTFORMAT
```

```
  'com.linkedin.haivvreo.AvroContainerOutputFormat';
```

The `schema.url` is a URL (here a `file://` URL) pointing to an Avro schema file that is used for reading and writing. It's also possible to specify a literal schema - see the documentation on GitHub for details.

When invoking Hive, pass the the `haivvreo`, `avro` and `avro-mapred` JARs in the `--auxpath` option to `hive`.

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`.

There are some additional notes on GitHub (the Hive Avro integration project, Haivvreo, was implemented by Jakob Homan from LinkedIn).

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-1.5.4.jar
```

See also RecordBreaker for information on turning text data into structured Avro data.

Mountable HDFS

CDH3 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-0.20-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 Ubuntu systems:

```
$ sudo apt-get install hadoop-0.20-fuse
```

To install `fuse-dfs` on Red Hat systems:

```
$ sudo yum install hadoop-0.20-fuse
```

To install `fuse-dfs` on SUSE systems:

```
$ sudo zypper install hadoop-0.20-fuse
```

You now have everything you need to begin mounting HDFS on Linux.

To set up and test your mount point:

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs
dfs://<name_node_hostname>:<namenode_port> <mount_point>
-d
```

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.

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.

Some options you can change:

- HADOOP_HOME can be changed. By default, it points to: `/usr/lib/hadoop`
- JAVA_HOME can be changed. By default, it points to: `/usr/java/default` and then `/usr/lib/jvm/java-6-sun`
- LD_LIBRARY_PATH can be changed. By default, it will track down `libjvm.so` in your `JAVA_HOME` and also contains `/usr/lib`.
- The JVM maximum heap size can be changed. By default, the CDH3 package installation creates the `/etc/default/hadoop-0.20-fuse` file with the following default JVM maximum heap size, which you can change:

```
export LIBHDFS_OPTS="-Xmx128m"
```

For more information, see the `man` pages for `hadoop-fuse-dfs`.

Java Development Kit Installation

You should install the Java Development Kit (JDK) before deploying CDH3; follow the instructions below.

> **Important**
>
> CDH3 requires JDK 1.6.0_8 at a minimum. Cloudera recommends versions 1.6.0_20 and 1.6.0_21; earlier versions are not recommended, and Cloudera has not yet tested later versions sufficiently to recommend them with full confidence.
>
> Before deploying CDH in a cluster, make sure you have the same version of the JDK on each node.

You may be able to install the JDK with your package manager, depending on your choice of operating system.

JDK Installation on Red Hat 5 and 6, CentOS 5 and 6, or SLES 11 Systems

RPM-based installation is available on Red Hat/CentOS 5 and 6, Red Hat 6, and SLES 11 Systems.

> **Important**
>
> The Oracle JDK installer is available both as an RPM-based installer (note the "`-rpm`" modifier before the `bin` file extension) for RPM-based systems, and as a binary installer for other systems. Make sure you install the `jdk-6uXX-linux-x64-rpm.bin` file for 64-bit systems and `jdk-6uXX-linux-i586-rpm.bin` for 32-bit 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 JDK:

1. Download one of the recommended versions of the JDK from this page, which you can also reach by going to the Java SE Downloads page, clicking on the **Previous Releases** tab, then on the **Archived Releases** link, then on the **Java SE 6** link. (These links and directions were correct at the time of writing, but the page is restructured frequently.)

2. Install the JDK following the directions on the the Java SE Downloads page.

JDK Installation on Ubuntu Systems

In the following example, replace *RELEASE* with the name of your release, which you can find by running `lsb_release -c`.

```
$ sudo add-apt-repository "deb
http://archive.canonical.com/ RELEASE partner"
$ sudo apt-get update
$ sudo sudo apt-get install sun-java6-jdk
```

You can press the **Tab** key to navigate to the **OK** button on the text box raised after the last command, and then press the **Spacebar** to select **OK**.

Creating a Local Yum Respository

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:

1. 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. From a computer that *does* have Internet access, download the yum repository into a temporary location. On Red Hat/CentOS 6, you can use a command such as:

```
reposync --source  -r cloudera-cdh3
```

3. Put all the RPMs into a directory served by your web server, for example `/var/www/html/cdh/3/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/3/RPMS/`).
4. On your web server, go to `/var/www/html/cdh/3/` 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`).
5. Edit the repo file you got from Cloudera and replace the line starting with `mirrorlist=` with `baseurl=http://<yourwebserver>/cdh/3/`
6. Put this modified repo file into `/etc/yum.repos.d/` on one of your machines, and check that you can install CDH through yum.
   Example:

```
yum update && yum install hadoop-0.20
```

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.

Using the CDH3 Maven Repository

If you want to build applications or tools with the CDH3 components and you are using Maven or Ivy for dependency management, you can pull the CDH3 artifacts from the Cloudera Maven repository. The repository is available at https://repository.cloudera.com/artifactory/cloudera-repos/.

The following table lists the groupId, artifactId, and version required to access each CDH3 artifact.

groupId	artifactId	version
org.apache.hadoop	hadoop-core	0.20.2-cdh3u2
org.apache.hadoop	hadoop-examples	0.20.2-cdh3u2
org.apache.hadoop	hadoop-streaming	0.20.2-cdh3u2
org.apache.hadoop	hadoop-tarball	0.20.2-cdh3u2
org.apache.hadoop	hadoop-mrunit	0.20.2-cdh3u2
org.apache.hadoop	hadoop-test	0.20.2-cdh3u2
org.apache.hadoop	hadoop-tools	0.20.2-cdh3u2
org.apache.hadoop.hive	hive-anttasks	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-cli	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-common	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-contrib	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-exec	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-hwi	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-jdbc	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-metastore	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-serde	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-service	0.7.1-cdh3u2
org.apache.hadoop.hive	hive-shims	0.7.1-cdh3u2
org.apache.hbase	hbase	0.90.4-cdh3u2

groupId	artifactId	version
org.apache.mahout	mahout-buildtools	0.5-cdh3u2
org.apache.mahout	mahout-core	0.5-cdh3u2
org.apache.mahout	mahout-distribution	0.5-cdh3u2
org.apache.mahout	mahout-eclipse-support	0.5-cdh3u2
org.apache.mahout	mahout-examples	0.5-cdh3u2
org.apache.mahout	mahout-math	0.5-cdh3u2
org.apache.mahout	mahout-taste-webapp	0.5-cdh3u2
org.apache.mahout	mahout-utils	0.5-cdh3u2
org.apache.pig	pig	0.8.1-cdh3u2
org.apache.whirr	whirr	0.5.0-cdh3u2
org.apache.whirr	whirr-build-tools	0.5.0-cdh3u2
org.apache.whirr	whirr-cassandra	0.5.0-cdh3u2
org.apache.whirr	whirr-cdh	0.5.0-cdh3u2
org.apache.whirr	whirr-cli	0.5.0-cdh3u2
org.apache.whirr	whirr-core	0.5.0-cdh3u2
org.apache.whirr	whirr-hadoop	0.5.0-cdh3u2
org.apache.whirr	whirr-hbase	0.5.0-cdh3u2
org.apache.whirr	whirr-zookeeper	0.5.0-cdh3u2
org.apache.zookeeper	zookeeper	3.3.3-cdh3u2

groupId	artifactId	version
com.cloudera.flume	flume	0.9.4-cdh3u2
com.cloudera.sqoop	sqoop	1.3.0-cdh3u2
com.yahoo.oozie	oozie	2.3.2-cdh3u2
com.yahoo.oozie	oozie-client	2.3.2-cdh3u2
com.yahoo.oozie	oozie-core	2.3.2-cdh3u2
com.yahoo.oozie	oozie-docs	2.3.2-cdh3u2
com.yahoo.oozie	oozie-examples	2.3.2-cdh3u2
com.yahoo.oozie	oozie-main	2.3.2-cdh3u2
com.yahoo.oozie	oozie-sharelib	2.3.2-cdh3u2
com.yahoo.oozie	oozie-webapp	2.3.2-cdh3u2

Building RPMs from CDH Source RPMs

This section describes how to build binary packages (RPMs) from published CDH source packages (SRPMs).

Prerequisites

- 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
```

SUSE 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 CDH3 reside at http://archive.cloudera.com/redhat/cdh/3/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

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.

For more information, see: http://www.cloudera.com/hadoop-support

Community Support

If you have any questions or comments about Cloudera's Distribution including Apache Hadoop (CDH), you can also send a message to the CDH user's list: cdh-user@cloudera.org

Apache and Third-Party Licenses

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.

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at:

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Third-Party Licenses

For a list of third-party licenses associated with CDH, see Third-Party Licenses.