

# CDH4 Installation Guide



## **Important Notice**

(c) 2010-2014 Cloudera, Inc. All rights reserved.

Cloudera, the Cloudera logo, Cloudera Impala, and any other product or service names or slogans contained in this document are trademarks of Cloudera and its suppliers or licensors, and may not be copied, imitated or used, in whole or in part, without the prior written permission of Cloudera or the applicable trademark holder.

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

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

Cloudera may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Cloudera, the furnishing of this document does not give you any license to these patents, trademarks copyrights, or other intellectual property.

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

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

## **Release Information**

Version: 4.7.0  
Date: May 29, 2014

# Table of Contents

<b>About this Guide .....</b>	<b>15</b>
CDH4 Components.....	15
Other Topics in this Guide.....	16
 <b>What's New in CDH4.....</b>	 <b>17</b>
 <b>Before You Install or Upgrade CDH4 on a Cluster .....</b>	 <b>19</b>
 <b>CDH4 Installation.....</b>	 <b>21</b>
CDH4 and MapReduce .....	21
<i>MapReduce 2.0 (YARN)</i> .....	21
Installing CDH4.....	22
<i>Ways To Install CDH4</i> .....	22
<i>Before You Begin Installing CDH4 Manually</i> .....	22
<i>Steps for Installing CDH4 Manually</i> .....	23
Installing CDH4 Components.....	32
Viewing the Apache Hadoop Documentation.....	33
 <b>Installing an Earlier CDH4 Release.....</b>	 <b>35</b>
Downloading and Installing an Earlier Release.....	35
<i>On Red Hat-compatible systems</i> .....	35
<i>On SLES systems</i> .....	36
<i>On Ubuntu and Debian systems</i> .....	37
 <b>Upgrading from CDH3 to CDH4.....</b>	 <b>39</b>
CDH4 and MapReduce.....	39
<i>MapReduce 2.0 (YARN)</i> .....	39
<i>High Availability</i> .....	40
Before You Begin.....	40
<i>Plan Downtime</i> .....	40
<i>Considerations for Secure Clusters</i> .....	40
Upgrading to CDH4.....	40
<i>Step 1: Back Up Configuration Data and Uninstall Components</i> .....	41
<i>Step 2: Back up the HDFS Metadata</i> .....	42

Step 3: Copy the Hadoop Configuration to the Correct Location and Update Alternatives.....	42
Step 4: Uninstall CDH3 Hadoop.....	43
Step 5: Download CDH4.....	44
Step 6a: Install CDH4 with MRv1.....	46
Step 6b: Install CDH4 with YARN.....	47
Step 7: Copy the CDH4 Logging File.....	49
Step 7a: (Secure Clusters Only) Set Variables for Secure DataNodes.....	49
Step 8: Upgrade the HDFS Metadata.....	49
Step 9: Create the HDFS /tmp Directory.....	50
Step 10: Start MapReduce (MRv1) or YARN.....	50
Step 11: Set the Sticky Bit.....	54
Step 12: Re-Install CDH4 Components.....	54
Step 13: Apply Configuration File Changes.....	55
Step 14: Finalize the HDFS Metadata Upgrade.....	55

## **Migrating data between a CDH3 and CDH4 cluster.....57**

Requirements.....	57
Using DistCp to Migrate Data between two Clusters.....	57
The DistCp Command.....	57
Post-migration Verification.....	58

## **Upgrading from an Earlier CDH4 Release .....59**

Before You Begin.....	59
Upgrading to the Latest Version of CDH4.....	60
Step 1: Prepare the cluster for the upgrade.....	60
Step 2: Download the CDH4 package on each of the hosts in your cluster.....	61
Step 3: Upgrade the packages on the appropriate hosts.....	62
Step 4: Upgrade the HDFS Metadata (Beta 1 or earlier).....	66
Step 5: Start HDFS (Beta 2 or later).....	67
Step 5a: Verify that /tmp Exists and Has the Right Permissions.....	67
Step 6: Start MapReduce (MRv1) or YARN.....	68
Step 7: Set the Sticky Bit.....	72
Step 8: Upgrade Components to CDH4.....	72
Step 9: Apply Configuration File Changes.....	73
Step 10: Finalize the HDFS Metadata Upgrade (Beta 1 or earlier).....	73

## **Configuring Ports for CDH4.....75**

Ports Used by Components of CDH4.....	75
Ports Used by Third-Party Components.....	79

## **Deploying CDH4 in Pseudo-Distributed Mode.....81**

## Deploying CDH4 on a Cluster .....83

Configuring Network Names.....	83
Deploying HDFS on a Cluster .....	84
Copying the Hadoop Configuration .....	84
Customizing Configuration Files .....	85
Configuring Local Storage Directories .....	86
Configuring DataNodes to Tolerate Local Storage Directory Failure .....	88
Formatting the NameNode .....	89
Configuring a Remote NameNode Storage Directory .....	89
Configuring the Secondary NameNode .....	90
Enabling Trash .....	91
Configuring Storage-Balancing for the DataNodes.....	93
Enabling WebHDFS .....	93
Configuring LZO .....	93
Deploy MRv1 or YARN .....	94
Deploying MapReduce v1 (MRv1) on a Cluster.....	94
Step 1: Configuring Properties for MRv1 Clusters.....	95
Step 2: Configure Local Storage Directories for Use by MRv1 Daemons.....	95
Step 3: Configure a Health Check Script for DataNode Processes.....	96
Step 4: Configure JobTracker Recovery.....	97
Enabling JobTracker Recovery.....	97
Step 5: Deploy your Custom Configuration to your Entire Cluster.....	97
Step 6: Start HDFS on Every Node in the Cluster.....	98
Step 7: Create the HDFS /tmp Directory.....	98
Step 8: Create MapReduce /var directories.....	98
Step 9: Verify the HDFS File Structure.....	98
Step 10: Create and Configure the mapred.system.dir Directory in HDFS.....	98
Step 11: Start MapReduce.....	99
Step 12: Create a Home Directory for each MapReduce User.....	99
Configure the Hadoop Daemons to Start at Boot Time.....	99
Deploying MapReduce v2 (YARN) on a Cluster.....	99
About MapReduce v2 (YARN).....	100
Step 1: Configure Properties for YARN Clusters.....	101
Step 2: Configure YARN daemons.....	101
Step 3: Configure the History Server.....	104
Step 4: Configure the Staging Directory.....	104
Step 5: Deploy your custom Configuration to your Entire Cluster.....	105
Step 6: Start HDFS on Every Node in the Cluster.....	105
Step 7: Create the HDFS /tmp Directory.....	105
Step 8: Create the history Directory and Set Permissions and Owner.....	106
Step 9: Create Log Directories.....	106
Step 10: Verify the HDFS File Structure.....	106
Step 11: Start YARN and the MapReduce JobHistory Server.....	106

<i>Step 12: Create a Home Directory for each MapReduce User</i> .....	107
<i>Step 13: Set HADOOP_MAPRED_HOME</i> .....	107
<i>Step 14: Configure the Hadoop Daemons to Start at Boot Time</i> .....	107
Configuring the Daemons to Start in an MRv1 Cluster.....	107
Tips and Guidelines.....	107
<i>Selecting Appropriate JAR files for your MRv1 and YARN Jobs</i> .....	107
<i>Improving Performance</i> .....	108
<i>Setting Quotas</i> .....	112

## Flume Installation.....113

Migrating to Flume 1.x from Flume 0.9.x.....	113
<i>Step 1: Remove Flume 0.9.x from your cluster</i> .....	114
<i>Step 2: Install the new version of Flume</i> .....	114
Flume Packaging.....	114
Installing the Flume Tarball.....	114
Installing the Flume RPM or Debian Packages.....	115
Flume Configuration.....	116
Verifying the Installation.....	117
Running Flume.....	117
Files Installed by the Flume RPM and Debian Packages.....	118
Supported Sources, Sinks, and Channels.....	118
<i>Sources</i> .....	118
<i>Sinks</i> .....	120
<i>Channels</i> .....	121
Using an On-disk Encrypted File Channel.....	121
<i>Generating Encryption Keys</i> .....	121
<i>Configuration</i> .....	122
<i>Changing Encryption Keys Over Time</i> .....	122
<i>Troubleshooting</i> .....	123
Viewing the Flume Documentation.....	124

## Sqoop Installation.....125

Upgrading Sqoop from CDH3 to CDH4.....	125
<i>Step 1: Remove the CDH3 version of Sqoop</i> .....	125
<i>Step 2: Install the new version of Sqoop</i> .....	126
Upgrading Sqoop from an Earlier CDH4 release.....	126
Sqoop Packaging.....	126
Sqoop Prerequisites.....	126
Installing the Sqoop RPM or Debian Packages.....	127
Installing the Sqoop Tarball.....	127
Installing the JDBC Drivers.....	128

<i>Installing the MySQL JDBC Driver.....</i>	<i>128</i>
<i>Installing the Oracle JDBC Driver.....</i>	<i>128</i>
<i>Installing the Microsoft SQL Server JDBC Driver.....</i>	<i>128</i>
<i>Installing the PostgreSQL JDBC Driver.....</i>	<i>129</i>
Setting HADOOP_MAPRED_HOME for YARN.....	129
Viewing the Sqoop Documentation.....	129

## **Sqoop 2 Installation.....131**

About Sqoop 2.....	131
<i>Sqoop 2 Packaging.....</i>	<i>131</i>
Installing Sqoop 2.....	131
Configuring Sqoop 2.....	132
<i>Configuring which Hadoop Version to Use.....</i>	<i>132</i>
<i>Installing the JDBC Drivers.....</i>	<i>133</i>
Starting, Stopping, and Accessing the Sqoop 2 Server.....	133
<i>Starting the Sqoop 2 Server.....</i>	<i>133</i>
<i>Stopping the Sqoop 2 Server.....</i>	<i>134</i>
<i>Checking that the Sqoop 2 Server has Started.....</i>	<i>134</i>
<i>Accessing the Sqoop 2 Server with the Sqoop 2 Client.....</i>	<i>134</i>
Viewing the Sqoop 2 Documentation.....	134

## **Hue Installation.....135**

Supported Browsers.....	136
Upgrading Hue.....	136
<i>Upgrading Hue from CDH3 to CDH4.....</i>	<i>136</i>
<i>Upgrading Hue from an Earlier CDH4 Release to the Latest CDH4 Release.....</i>	<i>137</i>
Installing Hue.....	137
<i>Installing the Hue Packages.....</i>	<i>137</i>
Configuring CDH Components for Hue.....	139
<i>WebHDFS or HttpFS Configuration.....</i>	<i>139</i>
<i>MRv1 Configuration.....</i>	<i>140</i>
<i>Oozie Configuration.....</i>	<i>142</i>
<i>Hive Configuration.....</i>	<i>142</i>
<i>Other Hadoop Settings.....</i>	<i>142</i>
Configuring Your Firewall for Hue.....	143
Hue Configuration.....	143
<i>Viewing the Hue Configuration.....</i>	<i>143</i>
<i>Hue Server Configuration.....</i>	<i>144</i>
<i>Beeswax Configuration.....</i>	<i>145</i>
<i>Cloudera Impala Query UI Configuration.....</i>	<i>145</i>
<i>Pig Editor Configuration.....</i>	<i>146</i>
<i>Sqoop Configuration.....</i>	<i>146</i>

<i>Job Browser Configuration</i> .....	146
<i>Job Designer</i> .....	146
<i>Oozie Editor/Dashboard Configuration</i> .....	146
<i>Search Configuration</i> .....	147
<i>Hue Shell Configuration</i> .....	147
<i>HBase Configuration</i> .....	149
<i>User Admin Configuration</i> .....	149
<i>Configuring an LDAP Server for User Admin</i> .....	149
<i>Hadoop Configuration</i> .....	152
<i>Liboozie Configuration</i> .....	153
Starting and Stopping the Hue Server.....	153
Administering Hue.....	153
<i>Anonymous Usage Data Collection</i> .....	154
<i>Processes</i> .....	154
<i>Logs</i> .....	154
<i>Hue Database</i> .....	154
Viewing the Hue User Guide.....	161

## **Pig Installation..... 163**

Upgrading Pig.....	163
<i>Upgrading Pig from CDH3 to CDH4</i> .....	163
<i>Upgrading Pig from an Earlier CDH4 release</i> .....	164
Installing Pig.....	164
Using Pig with HBase.....	166
Installing DataFu.....	166
Viewing the Pig Documentation.....	167

## **Oozie Installation..... 169**

About Oozie.....	169
Oozie Packaging.....	169
Oozie Prerequisites.....	169
Upgrading Oozie.....	170
<i>Upgrading Oozie from CDH3 to CDH4</i> .....	170
<i>Upgrading Oozie from an Earlier CDH4 Release to the Latest CDH4 Release</i> .....	171
Installing Oozie.....	172
Configuring Oozie.....	173
<i>Configuring which Hadoop Version to Use</i> .....	173
<i>Configuring Oozie after Upgrading from CDH3</i> .....	173
<i>Configuring Oozie after Upgrading from an Earlier CDH4 Release</i> .....	176
<i>Configuring Oozie after a New Installation</i> .....	177
Starting, Stopping, and Accessing the Oozie Server.....	184
<i>Starting the Oozie Server</i> .....	184



<i>Stopping the Oozie Server.....</i>	184
<i>Accessing the Oozie Server with the Oozie Client.....</i>	185
<i>Accessing the Oozie Server with a Browser.....</i>	185
Configuring Oozie Failover ( hot/cold ).....	185
<i>Points to note.....</i>	186
Viewing the Oozie Documentation.....	186

## **Hive Installation.....187**

About Hive.....	187
<i>HiveServer2.....</i>	187
Upgrading Hive.....	188
<i>Upgrading Hive from CDH3 to CDH4.....</i>	188
<i>Upgrading Hive from an Earlier Version of CDH4.....</i>	190
Installing Hive.....	192
Configuring the Hive Metastore.....	193
<i>Metastore Deployment Modes.....</i>	193
<i>Supported Metastore Databases.....</i>	194
<i>Configuring the Metastore Database.....</i>	194
Configuring HiveServer2.....	203
<i>Table Lock Manager (Required).....</i>	203
<i>hive.zookeeper.client.port.....</i>	204
<i>JDBC driver.....</i>	204
<i>Authentication.....</i>	204
<i>Configuring HiveServer2 for YARN.....</i>	205
<i>Running HiveServer2 and HiveServer Concurrently.....</i>	205
<i>Using Custom UDFs with HiveServer2.....</i>	205
Starting the Metastore.....	206
File System Permissions.....	206
Hive and HDFS Trash.....	206
Starting, Stopping, and Using HiveServer2.....	207
<i>Using the Beeline CLI.....</i>	207
Starting HiveServer1 and the Hive Console.....	208
Using Hive with HBase.....	208
Installing the Hive JDBC on Clients.....	209
Setting HADOOP_MAPRED_HOME for YARN.....	209
Configuring the Metastore to use HDFS High Availability.....	209
Troubleshooting.....	210
<i>Hive Queries Fail with "Too many counters" Error.....</i>	210
Viewing the Hive Documentation.....	210

## **Installing and Using HCatalog.....211**

HCatalog Prerequisites.....	211
Installing the HCatalog RPM or Debian Packages.....	211
<i>Installing the WebHCat REST Server.....</i>	212
<i>Installing HCatalog for Use with Pig and MapReduce.....</i>	212
Configuration Change on Hosts Used with HCatalog.....	212
Starting and Stopping the WebHCat REST server.....	213
Accessing Table Information with the HCatalog Command-line API.....	213
Accessing Table Data with MapReduce.....	213
Accessing Table Data with Pig .....	215
Accessing Table Information with REST.....	215
Viewing the HCatalog Documentation.....	216

## **HBase Installation.....217**

Upgrading HBase.....	217
<i>About Checksums in CDH4.2.....</i>	218
<i>Upgrading HBase from CDH3 to CDH4.....</i>	218
<i>Upgrading HBase from an Earlier CDH4 Release.....</i>	220
Installing HBase.....	221
Configuration Settings for HBase.....	222
<i>Using DNS with HBase.....</i>	222
<i>Using the Network Time Protocol (NTP) with HBase.....</i>	222
<i>Setting User Limits for HBase.....</i>	222
<i>Using dfs.datanode.max.xcievers with HBase.....</i>	223
Starting HBase in Standalone Mode.....	223
<i>Installing the HBase Master.....</i>	223
<i>Starting the HBase Master.....</i>	224
<i>Installing and Configuring REST.....</i>	224
Configuring HBase in Pseudo-Distributed Mode.....	225
<i>Modifying the HBase Configuration.....</i>	225
<i>Creating the /hbase Directory in HDFS.....</i>	225
<i>Enabling Servers for Pseudo-distributed Operation.....</i>	226
<i>Installing and Starting the HBase Thrift Server.....</i>	227
Deploying HBase in a Distributed Cluster.....	227
<i>Choosing Where to Deploy the Processes.....</i>	227
<i>Configuring for Distributed Operation.....</i>	228
Accessing HBase by using the HBase Shell.....	228
Using MapReduce with HBase.....	228
Troubleshooting.....	229
<i>Table Creation Fails after Installing LZO.....</i>	229
Viewing the HBase Documentation.....	229
HBase Replication.....	229
<i>Types of Replication.....</i>	229

<i>Points to Note about Replication.....</i>	<i>230</i>
<i>Requirements.....</i>	<i>230</i>
<i>Deploying HBase Replication.....</i>	<i>230</i>
<i>Disabling Replication at the Peer Level.....</i>	<i>232</i>
<i>Stopping Replication in an Emergency.....</i>	<i>232</i>
<i>Initiating Replication of Pre-existing Data.....</i>	<i>232</i>
<i>Verifying Replicated Data.....</i>	<i>233</i>
<i>Configuring Secure HBase Replication.....</i>	<i>233</i>
<i>Caveats.....</i>	<i>233</i>
Configuring HBase Snapshots.....	234
<i>About HBase Snapshots.....</i>	<i>234</i>
<i>Configuring and Enabling Snapshots.....</i>	<i>235</i>
<i>Shell Commands.....</i>	<i>236</i>
<i>Exporting a Snapshot to Another Cluster.....</i>	<i>238</i>
<i>Restrictions.....</i>	<i>238</i>
<i>Snapshot Failures.....</i>	<i>239</i>
<i>Information and Debugging.....</i>	<i>239</i>

## **ZooKeeper Installation.....241**

Upgrading ZooKeeper from CDH3 to CDH4.....	241
<i>Step 1: Remove ZooKeeper.....</i>	<i>242</i>
<i>Step 2: Install the ZooKeeper Base Package.....</i>	<i>242</i>
<i>Step 3: Install the ZooKeeper Server Package.....</i>	<i>242</i>
<i>Step 4: Edit /etc/zookeeper/conf/zoo.cfg or Move the Data Directory.....</i>	<i>243</i>
<i>Step 5: Restart the Server.....</i>	<i>243</i>
Upgrading ZooKeeper from an Earlier CDH4 Release.....	243
<i>Performing a ZooKeeper Rolling Upgrade.....</i>	<i>243</i>
Installing the ZooKeeper Packages.....	244
<i>Installing the ZooKeeper Base Package.....</i>	<i>244</i>
<i>Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server.....</i>	<i>245</i>
<i>Installing ZooKeeper in a Production Environment.....</i>	<i>246</i>
<i>Setting up Supervisory Process for the ZooKeeper Server.....</i>	<i>246</i>
Maintaining a ZooKeeper Server.....	246
Viewing the ZooKeeper Documentation.....	246

## **Whirr Installation.....247**

Upgrading Whirr.....	247
<i>Upgrading Whirr from CDH3 to CDH4.....</i>	<i>247</i>
<i>Upgrading Whirr from an Earlier CDH4 Release to the Latest CDH4 Release.....</i>	<i>249</i>
Installing Whirr.....	249
Generating an SSH Key Pair.....	249
Defining a Whirr Cluster.....	250

<i>MRv1 Cluster</i> .....	250
<i>YARN Cluster</i> .....	250
Launching a Cluster.....	251
<i>Running a Whirr Proxy</i> .....	251
<i>Running a MapReduce job</i> .....	251
<i>Destroying a cluster</i> .....	252
Viewing the Whirr Documentation.....	252

## **Snappy Installation.....253**

Upgrading Snappy to CDH4.....	253
Snappy Installation.....	253
Using Snappy for MapReduce Compression.....	253
Using Snappy for Pig Compression.....	254
Using Snappy for Hive Compression.....	254
Using Snappy compression in Sqoop Imports.....	254
Using Snappy Compression with HBase.....	255
Viewing the Snappy Documentation.....	255

## **Mahout Installation.....257**

Upgrading Mahout.....	257
<i>Upgrading Mahout from CDH3 to CDH4</i> .....	257
<i>Upgrading Mahout from an Earlier CDH4 Release to the Latest CDH4 Release</i> .....	258
Installing Mahout.....	258
The Mahout Executable.....	259
Getting Started with Mahout.....	259
The Mahout Documentation.....	259

## **HttpFS Installation.....261**

About HttpFS.....	261
HttpFS Packaging.....	261
HttpFS Prerequisites.....	262
Installing HttpFS.....	262
Configuring HttpFS.....	262
<i>Configuring the HDFS HttpFS Will Use</i> .....	263
<i>Configuring the HttpFS Proxy User</i> .....	263
<i>Configuring HttpFS with Kerberos Security</i> .....	263
Starting the HttpFS Server.....	263
Stopping the HttpFS Server.....	264
Using the HttpFS Server with curl.....	264

<b>Avro Usage.....</b>	<b>265</b>
Avro Data Files.....	265
Compression.....	265
Flume.....	265
Sqoop.....	266
MapReduce.....	266
Streaming.....	266
Pig.....	267
Hive.....	267
Avro Tools.....	269
 <b>Using the Parquet File Format with Impala, Hive, Pig, and MapReduce.....</b>	 <b>271</b>
Using Parquet Tables with Impala.....	271
Using Parquet Tables in Hive.....	272
Using Parquet Files in Pig.....	272
Using Parquet Files in MapReduce.....	273
Parquet File Interoperability.....	276
Examples of Java Programs to Read and Write Parquet Files.....	276
 <b>Sentry Installation.....</b>	 <b>279</b>
Installing Sentry.....	279
 <b>Maintenance Tasks and Notes.....</b>	 <b>281</b>
Starting CDH Services.....	281
Configuring init to Start Core Hadoop System Services.....	282
<i>Configuring init to Start Core Hadoop System Services in an MRv1 Cluster.....</i>	<i>282</i>
<i>Configuring init to Start Core Hadoop System Services in a YARN Cluster.....</i>	<i>283</i>
<i>Configuring init to Start Non-core Hadoop System Services.....</i>	<i>284</i>
Stopping Services.....	286
Uninstalling CDH Components.....	289
<i>Uninstalling from Red Hat, CentOS, and Similar Systems.....</i>	<i>289</i>
<i>Uninstalling from Debian and Ubuntu.....</i>	<i>290</i>
<i>Uninstalling from SLES.....</i>	<i>291</i>
<i>Additional clean-up.....</i>	<i>291</i>
SSH and HTTPS in the Hadoop Cluster.....	291
SSH.....	292
HTTPS.....	292
The HDFS Balancer.....	292

<b>Mountable HDFS.....</b>	<b>295</b>
<b>Java Development Kit Installation.....</b>	<b>297</b>
Oracle JDK Installation.....	297
<b>Creating a Local Yum Repository.....</b>	<b>299</b>
<b>Using the CDH4 Maven Repository.....</b>	<b>301</b>
<b>Building RPMs from CDH Source RPMs.....</b>	<b>303</b>
Prerequisites.....	303
Setting up an environment for building RPMs.....	303
<i>Red Hat or CentOS systems.....</i>	<i>303</i>
<i>SLES systems.....</i>	<i>303</i>
Building an RPM.....	303
<b>Getting Support.....</b>	<b>305</b>
Cloudera Support.....	305
Community Support.....	305
Report Issues.....	305
Get Announcements about New CDH and Cloudera Manager Releases.....	305
<b>Apache and Third-Party Licenses.....</b>	<b>307</b>
Apache License.....	307
Third-Party Licenses.....	307

## About this Guide

This *CDH4 Installation Guide* is for Apache Hadoop developers and system administrators interested in Hadoop installation. It describes how to install and configure version 4 of Cloudera's Distribution Including Apache Hadoop (CDH4), and how to deploy it on a cluster.

The guide covers the following major topics.

### Before You Start:

- [What's New in CDH4](#) on page 17
- [Before You Install or Upgrade CDH4 on a Cluster](#) on page 19

### Installation tasks:

- [Install CDH4](#). Start here for a new installation on a cluster.
- [Deploy CDH4](#). Do these tasks after installing core Hadoop.
- Install [components](#). Install additional components after installing and deploying HDFS and MapReduce.

- **Note:** To install a release earlier than the current CDH4 release (for example if you want to add new nodes to a cluster without upgrading the cluster to the latest release), follow [these instructions](#).

### Upgrade tasks:

- [Upgrade from CDH3 to CDH4](#). Use these instructions if you are currently running a CDH3 release; *or*
- [Upgrade from an Earlier CDH4 release](#). Use these instructions if you are currently running a CDH4 release.
- Upgrade [components](#). Upgrade all installed components after upgrading core Hadoop.

- **Note:** Use [these instructions](#) to migrate data from a CDH3 cluster to a CDH4 cluster.

## CDH4 Components

Use the following sections to install or upgrade CDH4 components:

- [Flume](#)
- [Sqoop](#)
- [Sqoop2](#)
- [Hue](#)
- [Pig](#)
- [Oozie](#)
- [Hive](#)
- [HCatalog](#)
- [HBase](#)
- [ZooKeeper](#)
- [Whirr](#)
- [Snappy](#)
- [Mahout](#)
- [HttpFS](#)
- [Avro](#)

## Other Topics in this Guide

This guide also contains information on the following:

- [Configuring Ports for CDH4](#)
- [Maintenance Tasks and Notes](#)
- [Java Development Kit \(JDK\) Installation](#)
- [Mountable HDFS](#)
- [Creating a local Yum Repository](#)
- [Building RPMs from Source RPMs](#)
- [Getting Support](#) on page 305Getting Support
- [Apache and Third-Party Licenses](#)



## What's New in CDH4

CDH4 contains many changes and enhancements when compared to previous releases. For details about the changes in CDH4, see the [CDH4 Release Notes](#), and specifically the section [New Features](#). For links to the detailed change lists that describe the bug fixes and improvements to all of the CDH4 projects, see the packaging section of [CDH Version and Packaging Information](#).



## Before You Install or Upgrade CDH4 on a Cluster

- **Note: Running Services**

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

- **Important:**

- **Upgrading from CDH3:** If you are upgrading from CDH3, you must first uninstall CDH3, then install CDH4; see [Upgrading from CDH3 to CDH4](#).

Before you install CDH4 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 CDH4. See [CDH4 Requirements and Supported Versions](#).
2. If you haven't already done so, install the Oracle Java Development Kit. For instructions and recommendations, see [Java Development Kit Installation](#).

- **Important:**

On SLES 11 platforms, do not install or try to use the IBM Java version bundled with the SLES distribution; Hadoop will not run correctly with that version. Install the Oracle JDK following directions under [Java Development Kit Installation](#).



## CDH4 Installation

This section describes how to install CDH4. See the following sections for more information and instructions:

- [CDH4 and MapReduce](#)
- [Installing CDH4](#)
- [Installing CDH4 Components](#)
- [Apache Hadoop Documentation](#)

## CDH4 and MapReduce

CDH4 introduces a new version of MapReduce: MapReduce 2.0 (MRv2) built on the YARN framework. In this document we usually refer to this new version as **YARN**. CDH4 also provides an implementation of the previous version of MapReduce, now referred to as **MRv1**. You can use the [instructions in the next section](#) to install:

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

▪ **Important:**

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

Before deciding to deploy YARN, make sure you read the discussion below under [MapReduce 2.0 \(YARN\)](#).

### MapReduce 2.0 (YARN)

MapReduce has undergone a complete overhaul and CDH4 now includes MapReduce 2.0 (MRv2). The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks. For details of the new architecture, see [Apache Hadoop NextGen MapReduce \(YARN\)](#).

- **Note:** Cloudera does not consider the current upstream MRv2 release stable yet, and it could potentially change in non-backwards-compatible ways. Cloudera recommends that you use MRv1 unless you have particular reasons for using MRv2, which should not be considered production-ready.

For more information about the two implementations (MRv1 and MRv2) see the discussion under Apache Hadoop MapReduce in the "What's New in Beta 1" section of [New Features in CDH4](#).

See also [Selecting Appropriate JAR files for your MRv1 and YARN Jobs](#).

# Installing CDH4

This section describes the process for installing CDH4.

## Ways To Install CDH4

You can install CDH4 in any of the following ways:

- Automated method using Cloudera Manager; instructions [here](#). Cloudera Manager automates the installation and configuration of CDH4 on an entire cluster if you have root or password-less `sudo` SSH access to your cluster's machines.

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

- Manual methods described under [Installing CDH4](#) on page 22:
  - Download and install the CDH4 "1-click Install" package
  - Add the CDH4 repository
  - Build your own CDH4 repository

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

- Install from a CDH4 tarball — see [How Packaging Affects CDH4 Deployment](#).

## How Packaging Affects CDH4 Deployment

### Installing from Packages

- To install and deploy MRv1, follow the directions on this page and then proceed with [Deploying MapReduce v1 \(MRv1\) on a Cluster](#).
- To install and deploy YARN, read the discussion under [New Features](#), then follow the directions on this page and proceed with [Deploying MapReduce v2 \(YARN\) on a Cluster](#).

### Installing from a Tarball

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

- If you install CDH4 from a [tarball](#), you will install YARN. Read the discussion of YARN under [New Features](#) before you proceed.
- As of CDH4.3.0, there is no separate tarball for MRv1. Instead, the MRv1 binaries, examples, etc., are delivered in the Hadoop tarball itself. The scripts for running MRv1 are in the `bin-mapreduce1` directory in the tarball, and the MRv1 examples are in the `examples-mapreduce1` directory.

## Before You Begin Installing CDH4 Manually

- This section contains instructions for new installations. If you need to upgrade from an earlier release, see [Upgrading from CDH3 to CDH4](#) or [Upgrading from an Earlier CDH4 Release](#).
- For a list of supported operating systems, see [CDH4 Requirements and Supported Versions](#).
- These instructions assume that the `sudo` command is configured on the hosts where you will be doing the installation. If this is not the case, you will need the root user (superuser) to configure it.

■ **Note: Running Services**

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

■ **Important:**

- **Java Development Kit:** if you have not already done so, install the Oracle Java Development Kit (JDK); see [Java Development Kit Installation](#).
- **Scheduler defaults:** note the following differences between MRv1 and MRv2 (YARN).
  - MRv1:
    - Cloudera Manager sets the default to FIFO.
    - CDH 4 sets the default to FIFO, with FIFO, Fair Scheduler, and Capacity Scheduler on the classpath by default.
  - MRv2 (YARN):
    - Cloudera Manager sets the default to Fair Scheduler.
    - CDH 4 sets the default to Fair Scheduler, with FIFO and Fair Scheduler on the classpath by default.
    - YARN does not support Capacity Scheduler.

## Steps for Installing CDH4 Manually

### Step 1: Add or Build the CDH4 Repository or Download the "1-click Install" package.

- If you are installing CDH4 on a [Red Hat](#) system, you can download Cloudera packages using `yum` or your web browser.
- If you are installing CDH4 on a [SLES](#) system, you can download the Cloudera packages using `zypper` or `YaST` or your web browser.
- If you are installing CDH4 on an [Ubuntu or Debian](#) system, you can download the Cloudera packages using `apt` or your web browser.

### On Red Hat-compatible Systems

Use one of the following methods to add or build the CDH4 repository or download the package on Red Hat-compatible systems:

■ **Note:**

Use only one of the three methods.

- [Download and install the CDH4 "1-click Install" package](#) *OR*
- [Add the CDH4 repository](#) *OR*
- [Build a Yum Repository](#)

Do this on all the systems in the cluster.

To download and install the CDH4 "1-click Install" package:

## CDH4 Installation

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/Oracle 5	<a href="#">Red Hat/CentOS/Oracle 5 link</a>
Red Hat/CentOS 6 (32-bit)	<a href="#">Red Hat/CentOS 6 link (32-bit)</a>
Red Hat/CentOS/Oracle 6 (64-bit)	<a href="#">Red Hat/CentOS/Oracle 6 link (64-bit)</a>

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

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

For Red Hat/CentOS 6 (32-bit):

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.i386.rpm
```

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

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

### **OR: To add the CDH4 repository:**

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

For OS Version	Click this Link
Red Hat/CentOS/Oracle 5	<a href="#">Red Hat/CentOS/Oracle 5 link</a>
Red Hat/CentOS 6 (32-bit)	<a href="#">Red Hat/CentOS 6 link</a>
Red Hat/CentOS/Oracle 6 (64-bit)	<a href="#">Red Hat/CentOS/Oracle 6 link</a>

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

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

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

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

## On SLES Systems

Use one of the following methods to download the CDH4 repository or package on SLES systems:

### ■ **Note:**

Use only one of the three methods.

- [Download and install the CDH4 "1-click Install" Package](#) *OR*
- [Add the CDH4 repository](#) *OR*



- [Build a SLES Repository](#)

To download and install the CDH4 "1-click Install" 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 cloudera-cdh-4-0.x86_64.rpm
```

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

**OR: To add the CDH4 repository:**

1. Run the following command:

```
$ sudo zypper addrepo -f
http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/cloudera-cdh4.repo
```

2. Update your system package index by running:

```
$ sudo zypper refresh
```

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

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

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

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

## On Ubuntu or Debian Systems

Use one of the following methods to download the CDH4 repository or package.

### Note:

Use only one of the three methods.

- [Download and install the CDH4 "1-click Install" Package](#) *OR*
- [Add the CDH4 repository](#) *OR*
- [Build a Debian Repository](#)

To download and install the CDH4 "1-click Install" package:

1. Click one of the following: [this link for a Squeeze system](#), or [this link for a Lucid system](#), or [this link for a Precise 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 cdh4-repository_1.0_all.deb
```

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

**OR: To add the CDH4 repository:**

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

- For Ubuntu systems:

```
deb [arch=amd64] http://archive.cloudera.com/cdh4/<OS-release-arch><RELEASE>-cdh4
contrib
deb-src http://archive.cloudera.com/cdh4/<OS-release-arch><RELEASE>-cdh4 contrib
```

- For Debian systems:

```
deb http://archive.cloudera.com/cdh4/<OS-release-arch><RELEASE>-cdh4 contrib
deb-src http://archive.cloudera.com/cdh4/<OS-release-arch><RELEASE>-cdh4 contrib
```

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

For example, to install CDH4 for 64-bit Ubuntu Lucid:

```
deb [arch=amd64] http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh lucid-cdh4
contrib
deb-src http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh lucid-cdh4 contrib
```

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

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

If you want to create your own apt repository, create a mirror of [the CDH Debian directory](#) and then [create an apt repository from the mirror](#).

Now continue with [Step 1a: Optionally Add a Repository Key](#), and then choose [Step 2: Install CDH4 with MRv1](#), or [Step 3: Install CDH4 with YARN](#); or do both steps if you want to install both implementations.

### Step 1a: Optionally Add a Repository Key

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

- For Red Hat/CentOS/Oracle 5 systems:

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- For Red Hat/CentOS/Oracle 6 systems:

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- For all SLES systems:

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- For Ubuntu Lucid systems:

```
$ curl -s http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Ubuntu Precise systems:**

```
$ curl -s http://archive.cloudera.com/cdh4/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Debian Squeeze systems:**

```
$ curl -s http://archive.cloudera.com/cdh4/debian/squeeze/amd64/cdh/archive.key
| sudo apt-key add -
```

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

## Step 2: Install CDH4 with MRv1

- **Note:**

Skip this step and go to [Step 3](#) if you intend to use *only* YARN.

- **Important:**

Before proceeding, you need to decide:

1. Whether to configure High Availability (HA) for the NameNode and/or JobTracker; see the [CDH4 High Availability Guide](#) for more information and instructions.
2. Where to deploy the NameNode, Secondary NameNode, and JobTracker daemons. As a general rule:
  - The NameNode and JobTracker run on the the same "master" host unless the cluster is large (more than a few tens of nodes), and the master host (or hosts) should not run the Secondary NameNode (if used), DataNode or TaskTracker services.
  - In a large cluster, it is especially important that the Secondary NameNode (if used) runs on a separate machine from the NameNode.
  - Each node in the cluster **except the master host(s)** should run the DataNode and TaskTracker services.

If you configure [HA for the NameNode](#), do not install `hadoop-hdfs-secondarynamenode`. After completing the software configuration for your [chosen HA method](#), follow the installation instructions under [HDFS High Availability Initial Deployment](#).

1. Install and deploy ZooKeeper.

- **Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

Where to install	Install commands
<b>JobTracker host</b> running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-jobtracker</code>

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

### Step 3: Install CDH4 with YARN

■ **Note:**

Skip this step if you intend to use *only* MRv1. Directions for installing MRv1 are in [Step 2](#).

**To install CDH4 with YARN:**

■ **Note:**

- If you are also installing MRv1, you can skip any packages you have already installed in [Step 2](#).
- If you configure [HA for the NameNode](#), do not install `hadoop-hdfs-secondarynamenode`. After completing the software configuration for your [chosen HA method](#), follow the installation instructions under [HDFS High Availability Initial Deployment](#).

**1. Install and deploy ZooKeeper.**

■ **Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

Where to install	Install commands
<b>Resource Manager host</b> (analogous to MRv1 JobTracker) running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager</code>
<b>NameNode host</b> running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-hdfs-namenode</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode</code>
<b>Secondary NameNode host</b> (if used) running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode</code>

Where to install	Install commands
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode</code>
<b>All cluster hosts except the Resource Manager running:</b>	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<b>One host in the cluster running:</b>	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<b>All client hosts running:</b>	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-client</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-client</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-client</code>

■ **Note:**

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

## Step 4: (Optional) Install LZO

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

1. Add the repository on each node in the cluster.

Follow the instructions for your OS version:

For OS Version	Do this
Red Hat/CentOS/Oracle 5	Navigate to <a href="#">this link</a> and save the file in the <code>/etc/yum.repos.d/</code> directory.
Red Hat/CentOS 6	Navigate to <a href="#">this link</a> and save the file in the <code>/etc/yum.repos.d/</code> directory.
SLES	<ol style="list-style-type: none"> <li>1. Run the following command:  <pre>\$ sudo zypper addrepo -f http://archive.cloudera.com/gplextras/sles/11/x86_64/gplextras/ cloudera-gplextras4.repo</pre> </li> <li>2. Update your system package index by running:  <pre>\$ sudo zypper refresh</pre> </li> </ol>
Ubuntu or Debian	Navigate to <a href="#">this link</a> and save the file as <code>/etc/apt/sources.list.d/gplextras.list</code> . <div> <ul style="list-style-type: none"> <li>■ <b>Important:</b> Make sure you do not let the file name default to <code>cloudera.list</code>, as that will overwrite your existing <code>cloudera.list</code>.</li> </ul> </div>

2. Install the package on each node as follows:

For OS version	Install commands
Red Hat/CentOS compatible	<code>sudo yum install hadoop-lzo-cdh4</code>
SLES	<code>sudo zypper install hadoop-lzo-cdh4</code>
Ubuntu or Debian	<code>sudo apt-get update; sudo apt-get install hadoop-lzo-cdh4</code>

3. Continue with installing and deploying CDH. As part of the the deployment, you will need to do some additional configuration for LZO, as shown under [Configuring LZO](#) on page 93.

- **Important:** Make sure you do this configuration *after* you have [copied the default configuration files](#) to a custom location and set alternatives to point to it.

## Step 5: Deploy CDH and Install Components

Now proceed with:

- [deploying CDH4](#)
- [installing components](#).

- **Note:**  
To see what files and directories a given package installs on your system, and the permissions, run one of the following commands:

- For RPMs:

```
rpm -v -ql -p <package>.rpm
```
- For Ubuntu/Debian packages:

```
deb -c <package>.deb
```

## Installing CDH4 Components

In a new installation, follow the steps under [Installing CDH4](#) on page 22 and [Deploying CDH4 on a Cluster](#) on page 83 before installing the components described on this page.

You can install and run the following components with CDH4.0.0:

- **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.
- **Sqoop 2** — A server-based tool for transferring data between Hadoop and relational databases. You can use Sqoop 2 to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data with Hadoop MapReduce, and then export it back into an RDBMS.
- **HCatalog** — A tool that provides table data access for CDH components such as Pig and MapReduce.
- **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 CDH4 components, see the following sections:**

- **Flume.** For more information, see "Flume Installation" in this guide.
- **Sqoop.** For more information, see "Sqoop Installation" in this guide.
- **Sqoop 2.** For more information, see "Sqoop 2 Installation" in this guide.
- **HCatalog.** For more information, see "Installing and Using HCatalog" in this guide.
- **Hue.** For more information, see "Hue Installation" in this guide.
- **Pig.** For more information, see "Pig Installation" in this guide.
- **Oozie.** For more information, see "Oozie Installation" in this guide.
- **Hive.** For more information, see "Hive Installation" in this guide.
- **HBase.** For more information, see "HBase Installation" in this guide.
- **ZooKeeper.** For more information, "ZooKeeper Installation" in this guide.
- **Whirr.** For more information, see "Whirr Installation" in this guide.
- **Snappy.** For more information, see "Snappy Installation" in this guide.
- **Mahout.** For more information, see "Mahout Installation" in this guide.



## Viewing the Apache Hadoop Documentation

- For additional Apache Hadoop documentation, see <http://archive.cloudera.com/cdh4/cdh/4/hadoop>.
- For more information about YARN, see the Apache Hadoop NextGen MapReduce (YARN) page at <http://archive.cloudera.com/cdh4/cdh/4/hadoop/hadoop-yarn/hadoop-yarn-site/YARN.html>.



## Installing an Earlier CDH4 Release

Follow [these instructions](#) to install a release **earlier than the current CDH release**.

A common reason for doing this would be that you need to add new nodes to an existing cluster that is not running the most recent version of CDH. For example your cluster might be running CDH4.0.1 when the most recent release is CDH4.1.2; in this case, you will want to install CDH4.0.1 on the new nodes, not CDH4.1.2.

## Downloading and Installing an Earlier Release

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

- [On Red Hat-compatible systems](#)
- [On SLES systems](#)
- [On Ubuntu and Debian systems](#)

### On Red Hat-compatible systems

#### Step 1. Download and Save the Yum Repo File

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

For OS Version	Click this Link
Red Hat/CentOS/Oracle 5	<a href="#">Red Hat/CentOS/Oracle 5 link</a>
Red Hat/CentOS 6 (32-bit)	<a href="#">Red Hat/CentOS 6 link</a>
Red Hat/CentOS 6 (64-bit)	<a href="#">Red Hat/CentOS 6 link</a>

#### Step 2. Edit the Repo File

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

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

```
[cloudera-cdh4]
name=Cloudera's Distribution for Hadoop, Version 4
baseurl=http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4/
gpgkey = http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

If you want to install CDH4.0.1, for example, change

`baseurl=http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4/` to

`baseurl=http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4.0.1/`

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

```
[cloudera-cdh4]
name=Cloudera's Distribution for Hadoop, Version 4
baseurl=http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4.0.1/
gpgkey = http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
gpgcheck = 1
```

## Installing an Earlier CDH4 Release

### Step 3: Proceed with the Installation

1. Go to [Documentation for CDH 4 Releases](#).
2. Find the Installation Guide for your release; for example, for CDH4.0.0 or CDH4.0.1, scroll down to the link for the "CDH4.0.0 Installation Guide" and click on the link to the PDF.
3. Follow the instructions on the "CDH4 Installation" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH4 with MRv1 or YARN, and is usually Step 1a.)

## On SLES systems

### Step 1. Add the Cloudera Repo

1. Run the following command:

```
$ sudo zypper addrepo -f  
http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/cloudera-cdh4.repo
```

2. Update your system package index by running:

```
$ sudo zypper refresh
```

### Step 2. Edit the Repo File

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

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

```
[cloudera-cdh4]  
name=Cloudera's Distribution for Hadoop, Version 4  
baseurl=http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/4/  
gpgkey = http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera  
gpgcheck = 1
```

If you want to install CDH4.0.1, for example, change

`baseurl=http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/4/` to

`baseurl= http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/4.0.1/`.

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

```
[cloudera-cdh4]  
name=Cloudera's Distribution for Hadoop, Version 4  
baseurl=http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/4.0.1/  
gpgkey = http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera  
gpgcheck = 1
```

### Step 3: Proceed with the Installation

1. Go to [Documentation for CDH 4 Releases](#).
2. Find the Installation Guide for your release; for example, for CDH4.0.0 or CDH4.0.1, scroll down to the link for the "CDH4.0.0 Installation Guide" and click on the link to the PDF.
3. Follow the instructions on the "CDH4 Installation" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH4 with MRv1 or YARN, and is usually Step 1a.)

## On Ubuntu and Debian systems

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

### Step 1: Create the Repo File

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

- For Ubuntu systems:

```
deb [arch=amd64] http://archive.cloudera.com/cdh4/ <OS-release-arch> <RELEASE>-cdh4
contrib deb-src http://archive.cloudera.com/cdh4/ <OS-release-arch> <RELEASE>-cdh4
contrib
```

- For Debian systems:

```
deb http://archive.cloudera.com/cdh4/ <OS-release-arch> <RELEASE>-cdh4 contrib
deb-src http://archive.cloudera.com/cdh4/ <OS-release-arch> <RELEASE>-cdh4 contrib
```

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

Now replace `-cdh4` near the end of each line (before `contrib`) with the CDH release you need to install. Here are some examples using CDH4.0.0:

#### For 64-bit Ubuntu Lucid:

```
deb [arch=amd64] http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh lucid-cdh4.0.0
contrib
deb-src http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh lucid-cdh4.0.0 contrib
```

#### For 64-bit Ubuntu Precise:

```
deb [arch=amd64] http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh lucid-cdh4.0.0
contrib
deb-src http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh precise-cdh4.0.0 contrib
```

#### For Debian Squeeze:

```
deb http://archive.cloudera.com/cdh4/debian/squeeze/amd64/cdh squeeze-cdh4.0.0 contrib
deb-src http://archive.cloudera.com/cdh4/debian/squeeze/amd64/cdh squeeze-cdh4.0.0
contrib
```

### Step 2: Proceed with the Installation

- Go to [Documentation for CDH 4 Releases](#).
- Find the Installation Guide for your release; for example, for CDH4.0.0 or CDH4.0.1, scroll down to the link for the "CDH4.0.0 Installation Guide" and click on the link to the PDF.
- Follow the instructions on the "CDH4 Installation" page, starting with the instructions for optionally adding a repository key. (This comes immediately before the steps for installing CDH4 with MRv1 or YARN, and is usually Step 1a.)



## Upgrading from CDH3 to CDH4

- **Note:**

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

- If you are running Cloudera Manager 3.x, **you must upgrade Cloudera Manager to version 4 first**, as Cloudera Manager 3 cannot manage CDH4. Follow directions in the current [Cloudera Manager Installation Guide](#) (or the directions in the Installation Guide for the Cloudera Manager 4.x version you are using) to upgrade CDH3 to CDH4 in a Cloudera-managed deployment.

Use the following information and instructions to upgrade to the latest CDH4 release from a CDH3 release.:

- [CDH4 and MapReduce](#)
- [Before You Begin](#)
- [Upgrading to CDH4](#)

- **Important:** This involves uninstalling the CDH3 packages and installing the CDH4 packages.

If you are upgrading from an earlier CDH4 release, see [Upgrading from an Earlier CDH4 Release](#).

## CDH4 and MapReduce

CDH4 introduces a new version of MapReduce: MapReduce 2.0 (MRv2) built on the YARN framework. In this document we usually refer to this new version as **YARN**. CDH4 also provides an implementation of the previous version of MapReduce, now referred to as **MRv1**. You can use the instructions on this page to install:

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

- **Important:**

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

Before deciding to deploy YARN, make sure you read the discussion below under [MapReduce 2.0 \(YARN\)](#).

### MapReduce 2.0 (YARN)

MapReduce has undergone a complete overhaul and CDH4 now includes MapReduce 2.0 (MRv2). The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks. For details of the new architecture, see [Apache Hadoop NextGen MapReduce \(YARN\)](#).

## Upgrading from CDH3 to CDH4

- **Note:**

Cloudera does not consider the current upstream MRv2 release stable yet, and it could potentially change in non-backwards-compatible ways. Cloudera recommends that you use MRv1 unless you have particular reasons for using MRv2, which should not be considered production-ready.

For more information about the two implementations (MRv1 and MRv2) see the discussion under Apache Hadoop MapReduce in the "What's New in Beta 1" section of [New Features](#).

See also [Selecting Appropriate JAR files for your MRv1 and YARN Jobs](#).

## High Availability

In CDH4 you can configure high availability both for the NameNode and the JobTracker. For more information and instructions, see the [CDH4 High Availability Guide](#).

- **Important:**

If you configure [HA for the NameNode](#), do not install `hadoop-hdfs-secondarynamenode`. After completing the software configuration for your [chosen HA method](#), follow the installation instructions under [HDFS High Availability Initial Deployment](#).

## Before You Begin

- **Important:**

Before upgrading, be sure to read about the latest [Incompatible Changes](#) and [Known Issues and Work Arouns in CDH4](#) in the [CDH4 Release Notes](#).

## Plan Downtime

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

## Considerations for Secure Clusters

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

- Before starting the upgrade, read the [CDH4 Security Guide](#).
- Before shutting down Hadoop services, put the NameNode into safe mode and perform a `saveNamespace` operation; see the instructions on [backing up the metadata](#).
- Before re-starting HDFS, make sure you set the appropriate variables following these [instructions](#); otherwise the DataNodes will not start.

## Upgrading to CDH4

Use the instructions that follow to upgrade to CDH4.



■ **Note: Running Services**

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

## Step 1: Back Up Configuration Data and Uninstall Components

1. If security is enabled, do the following (see the [CDH3 Security Guide](#) for more information about CDH3 security):

- a. Put the NameNode into safe mode:

```
$ hadoop dfsadmin -safemode enter
```

- b. Perform a `saveNamespace` operation:

```
$ hadoop dfsadmin -saveNamespace
```

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

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

2. For each component you are using, back up configuration data, databases, and other important files, stop the component, then uninstall it. See the following sections for instructions:

■ **Note:**

At this point, you are only removing the components; do not install the new versions yet.

- [Removing Flume 0.9.x](#)
- [Removing Sqoop](#)
- [Removing Hue](#)
- [Removing Pig](#)
- [Removing Oozie](#)
- [Removing Hive](#)
- [Removing HBase](#)
- [Removing ZooKeeper](#)
- [Removing Whirr](#)
- [Removing Mahout](#)

3. Make sure the Hadoop services are shut down across your entire cluster by

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

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

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

### Step 2: Back up the HDFS Metadata

- **Important:**

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

To back up the HDFS metadata on the NameNode machine:

- **Note:**

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

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

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

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

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

- **Warning:**

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

### Step 3: Copy the Hadoop Configuration to the Correct Location and Update Alternatives

For CDH4, Hadoop looks for the cluster configuration files in a different location from the one used in CDH3, so you need to copy the configuration to the new location and reset the `alternatives` to point to it. Proceed as follows.

**On each node in the cluster:**

1. Copy the existing configuration to the new location, for example:

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

2. Update the alternatives, for example:

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

3. Verify that the operation succeeded:

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

## Step 4: Uninstall CDH3 Hadoop

- **Warning:**

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

### To uninstall CDH3 Hadoop:

Run this command on each host:

#### On Red Hat-compatible systems:

```
$ sudo yum remove hadoop-0.20 bigtop-utils
```

#### On SLES systems:

```
$ sudo zypper remove hadoop-0.20 bigtop-utils
```

#### On Ubuntu systems:

```
sudo apt-get purge hadoop-0.20 bigtop-utils
```

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

To uninstall the repository packages, run this command on each host:

#### On Red Hat-compatible systems:

```
$ sudo yum remove cloudera-cdh3
```

#### On SLES systems:

```
$ sudo zypper remove cloudera-cdh
```

#### On Ubuntu and Debian systems:

```
sudo apt-get remove cdh3-repository
```

- **Important:** On Ubuntu and Debian systems, you need to re-create the `/usr/lib/hadoop-0.20/` directory after uninstalling CDH3. Make sure you do this before you install CDH4:

```
$ sudo mkdir -p /usr/lib/hadoop-0.20/
```

### Step 5: Download CDH4

On Red Hat-compatible systems:

1. Download the CDH4 "1-click Install" Package:
2. 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/Oracle 5	<a href="#">Red Hat/CentOS/Oracle 5 link</a>
Red Hat/CentOS 6 (32-bit)	<a href="#">Red Hat/CentOS 6 link (32-bit)</a>
Red Hat/CentOS/Oracle 6 (64-bit)	<a href="#">Red Hat/CentOS/Oracle 6 link (64-bit)</a>

3. Install the RPM.

For Red Hat/CentOS/Oracle 5:

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

For Red Hat/CentOS 6 (32-bit):

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.i386.rpm
```

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

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

- **Note:**

For instructions on how to add a CDH4 yum repository or build your own CDH4 yum repository, see [Installing CDH4 On Red Hat-compatible systems](#).

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

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

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

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

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

## On SLES systems:

1. Download the CDH4 "1-click Install" Package:
2. Click [this link](#), choose **Save File**, and save it to a directory to which you have write access (it can be your home directory).
3. Install the RPM:

```
$ sudo rpm -i cloudera-cdh-4-0.x86_64.rpm
```

- **Note:**

For instructions on how to add a repository or build your own repository, see [Installing CDH4 on SLES Systems](#).

4. Update your system package index by running:

```
$ sudo zypper refresh
```

5. (Optionally) add a repository key on each system in the cluster. Add the Cloudera Public GPG Key to your repository by executing the following command:

- **For all SLES systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

## On Ubuntu and Debian systems:

1. Download the CDH4 "1-click Install" Package:
2. Click one of the following: [this link for a Squeeze system](#), or [this link for a Lucid system](#), or [this link for a Precise system](#).
3. 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 cdh4-repository_1.0_all.deb
```

- **Note:**

For instructions on how to add a repository or build your own repository, see [Installing CDH4 on Ubuntu Systems](#).

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

- **For Ubuntu Lucid systems:**

```
$ curl -s http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Ubuntu Precise systems:**

```
$ curl -s http://archive.cloudera.com/cdh4/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- For Debian Squeeze systems:

```
$ curl -s http://archive.cloudera.com/cdh4/debian/squeeze/amd64/cdh/archive.key | sudo apt-key add -
```

## Step 6a: Install CDH4 with MRv1

- **Note:**

Skip this step and go to [Step 6b](#) if you intend to use *only* YARN.

1. Install and deploy ZooKeeper.

- **Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

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

Where to install	Install commands
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-0.20-mapreduce-tasktracker hadoop-hdfs-datanode</code>
<b>All client hosts</b> , running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-client</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-client</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-client</code>

## Step 6b: Install CDH4 with YARN

- Note:**

Skip this step if you intend to use *only* MRv1. Directions for installing MRv1 are in [Step 6a](#).

### To install CDH4 with YARN:

- Note:**

If you are also installing MRv1, you can skip any packages you have already installed in [Step 6a](#).

### 1. Install and deploy ZooKeeper.

- Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

Where to install	Install commands
<b>Resource Manager host</b> (analogous to MRv1 JobTracker) running:	
<i>Red Hat/CentOS compatible</i>	<code>\$ sudo yum clean all; sudo yum install hadoop-yarn-resourcemanager</code>
<i>SLES</i>	<code>\$ sudo zypper clean --all; sudo zypper install hadoop-yarn-resourcemanager</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-yarn-resourcemanager</code>

Where to install	Install commands
<b>NameNode</b> host running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-hdfs-namenode</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-hdfs-namenode</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-hdfs-namenode</code>
<b>Secondary NameNode</b> host (if used) running:	
<i>Red Hat/CentOS compatible</i>	<code>sudo yum clean all; sudo yum install hadoop-hdfs-secondarynamenode</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-hdfs-secondarynamenode</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-hdfs-secondarynamenode</code>
<b>All cluster hosts except the Resource Manager</b> (analogous to MRv1 TaskTrackers) running:	
<i>Red Hat/CentOS compatible</i>	<code>\$ sudo yum clean all; sudo yum install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<i>SLES</i>	<code>\$ sudo zypper clean --all; sudo zypper install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<i>Ubuntu or Debian</i>	<code>\$ sudo apt-get update; sudo apt-get install hadoop-yarn-nodemanager hadoop-hdfs-datanode hadoop-mapreduce</code>
<b>One host in the cluster</b> running:	
<i>Red Hat/CentOS compatible</i>	<code>\$ sudo yum clean all; sudo yum install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<i>SLES</i>	<code>\$ sudo zypper clean --all; sudo zypper install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-mapreduce-historyserver hadoop-yarn-proxyserver</code>
<b>All client</b> hosts, running:	
<i>Red Hat/CentOS compatible</i>	<code>\$ sudo yum clean all; sudo yum install hadoop-client</code>
<i>SLES</i>	<code>sudo zypper clean --all; sudo zypper install hadoop-client</code>
<i>Ubuntu or Debian</i>	<code>sudo apt-get update; sudo apt-get install hadoop-client</code>

■ **Note:**

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



## Step 7: Copy the CDH4 Logging File

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

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

## Step 7a: (Secure Clusters Only) Set Variables for Secure DataNodes

### ■ Important:

You must do the following if you are upgrading a CDH3 cluster that has Kerberos security enabled. Otherwise, skip this step.

In order to allow DataNodes to start on a secure Hadoop cluster, you must set the following variables on all DataNodes in `/etc/default/hadoop-hdfs-datanode`.

```
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/var/lib/hadoop-hdfs
export HADOOP_SECURE_DN_LOG_DIR=/var/log/hadoop-hdfs
export JSVC_HOME=/usr/lib/bigtop-utils/
```

### ■ Note:

Depending on the version of Linux you are using, you may not have the `/usr/lib/bigtop-utils` directory on your system. If that is the case, set the `JSVC_HOME` variable to the `/usr/libexec/bigtop-utils` directory by using this command: `export JSVC_HOME=/usr/libexec/bigtop-utils`

## Step 8: Upgrade the HDFS Metadata

1. To upgrade the HDFS metadata, run the following command on the NameNode:

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

### ■ Note:

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

You can watch the progress of the upgrade by running:

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

Look for a line that confirms the upgrade is complete, such as:

```
/var/lib/hadoop-hdfs/cache/hadoop/dfs/<name> is complete
```

2. Start up the DataNodes:

On each DataNode:

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

3. Wait for NameNode to exit safe mode, and then start the Secondary NameNode (if used) and complete the cluster upgrade.
  - a. To check that the NameNode has exited safe mode, look for messages in the log file, or the NameNode's web interface, that say "...no longer in safe mode."

## Upgrading from CDH3 to CDH4

- b. To start the Secondary NameNode (if used), enter the following command on the Secondary NameNode host:

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

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

### Step 9: Create the HDFS /tmp Directory

■ **Important:**

If you do not create /tmp properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create /tmp yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

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

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

■ **Note:**

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) or `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

### Step 10: Start MapReduce (MRv1) or YARN

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

For MRv1	or For YARN
<a href="#">Start MRv1</a>	<a href="#">Start YARN and the MapReduce JobHistory Server</a>
<a href="#">Verify basic cluster operation</a>	<a href="#">Verify basic cluster operation</a>

#### Step 10a: Start MapReduce (MRv1)

■ **Important:**

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

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

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

On the JobTracker system:

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

Verify that the JobTracker and TaskTracker started properly.

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

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

### Verify basic cluster operation for MRv1.

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

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

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

Do the following steps as the user joe.

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

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

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

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

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

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

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

5. List the output files.

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

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

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

You have now confirmed your cluster is successfully running CDH4.

- **Important:**

If you have client hosts, make sure you also update them to CDH4, and upgrade the [components](#) running on those clients as well.

### Step 10b: Start MapReduce with YARN

- **Important:**

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

Before deciding to deploy YARN, make sure you read the discussion under [New Features](#).

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

For more information see [Deploying MapReduce v2 \(YARN\) on a Cluster](#).

Create a history directory and set permissions; for example:

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

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

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

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

Verify the directory structure, ownership, and permissions:

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

You should see:

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

To start YARN, start the ResourceManager and NodeManager services:

- **Note:**

Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

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

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

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

### To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

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

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

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

### Verify basic cluster operation for YARN.

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

■ **Note:**

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

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

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

Do the following steps as the user `joe`.

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

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

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

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

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

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

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

```
$ hadoop fs -ls
Found 2 items
```

## Upgrading from CDH3 to CDH4

```
drwxr-xr-x  - joe supergroup  0 2009-08-18 18:36 /user/joe/input
drwxr-xr-x  - joe supergroup  0 2009-08-18 18:38 /user/joe/output23
```

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

### 6. List the output files.

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

### 7. Read the results in the output file.

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

You have now confirmed your cluster is successfully running CDH4.

#### ■ Important:

If you have client hosts, make sure you also update them to CDH4, and upgrade the [components](#) running on those clients as well.

## Step 11: Set the Sticky Bit

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

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

## Step 12: Re-Install CDH4 Components

To install the CDH4 components, see the following sections:

- Flume. For more information, see "Flume Installation" in this guide.
- Sqoop. For more information, see "Sqoop Installation" in this guide.
- Sqoop 2. For more information, see "Sqoop 2 Installation" in this guide.
- HCatalog. For more information, see "Installing and Using HCatalog" in this guide.
- Hue. For more information, see "Hue Installation" in this guide.
- Pig. For more information, see "Pig Installation" in this guide.
- Oozie. For more information, see "Oozie Installation" in this guide.
- Hive. For more information, see "Hive Installation" in this guide.
- HBase. For more information, see "HBase Installation" in this guide.
- ZooKeeper. For more information, "ZooKeeper Installation" in this guide.
- Whirr. For more information, see "Whirr Installation" in this guide.
- Snappy. For more information, see "Snappy Installation" in this guide.
- Mahout. For more information, see "Mahout Installation" in this guide.

## Step 13: Apply Configuration File Changes

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

For example, if you have modified your CDH3 `zoo.cfg` configuration file (`/etc/zookeeper.dist/zoo.cfg`), RPM uninstall and re-install (using `yum remove`) renames and preserves a copy of your modified `zoo.cfg` as `/etc/zookeeper.dist/zoo.cfg.rpmsave`. You should compare this to the new `/etc/zookeeper/conf/zoo.cfg` and resolve any differences that should be carried forward (typically where you have changed property value defaults). Do this for each component you upgrade to CDH4.

## Step 14: Finalize the HDFS Metadata Upgrade

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

1. Make sure you are satisfied that the CDH4 upgrade has succeeded and everything is running smoothly. This could take a matter of days, or even weeks.

- **Warning:**

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

- **Note:**

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

2. Finalize the HDFS metadata upgrade: use one of the following commands, depending on whether Kerberos is enabled (see [Configuring Hadoop Security in CDH4](#)).

- If Kerberos is enabled:

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

- If Kerberos is not enabled:

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

- **Note:**

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





## Migrating data between a CDH3 and CDH4 cluster

You can migrate the data from a CDH3 (or any Apache Hadoop) cluster to a CDH4 cluster by using a tool that copies out data in parallel, such as the DistCp tool offered in CDH4. This can be useful if you are not planning to upgrade your CDH3 cluster itself at this point. The following sections provide information and instructions:

- [Requirements](#)
- [Using DistCp to Migrate Data between two Clusters](#)
- [Post-Migration Verification](#)

### Requirements

1. The CDH4 cluster must have a MapReduce service running on it. This may be MRv1 or YARN (MRv2).
2. All the MapReduce nodes in the CDH4 cluster should have full network access to all the nodes of the source cluster. This allows you to perform the copy in a distributed manner.

■ **Note:**

The term **source** refers to the CDH3 (or other Hadoop) cluster you want to migrate or copy data from; and **destination** refers to the CDH4 cluster.

### Using DistCp to Migrate Data between two Clusters

You can use the DistCp tool on the CDH4 cluster to initiate the copy job to move the data. Between two clusters running different versions of CDH, run the DistCp tool with `hftp://` as the source file system and `hdfs://` as the destination file system.

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

where `namenode-location` refers to the CDH3's NameNode hostname as defined by its config `fs.default.name` and 50070 is the NameNode's HTTP server port, as defined by the config `dfs.http.address`.

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

This refers to the CDH4's NameNode as defined by its configured `fs.defaultFS`.

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

### The DistCp Command

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

```
$ hadoop distcp
```

Run the DistCp copy by issuing a command such as the following on the CDH4 cluster:

- **Important:** Run the following DistCp commands on the destination cluster only, in this example, the CDH4 cluster.

```
$ hadoop distcp hftp://cdh3-namenode:50070/ hdfs://cdh4-nameservice/
```

## Migrating data between a CDH3 and CDH4 cluster

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

```
$ hadoop distcp hftp://cdh3-namenode:50070/hbase hdfs://cdh4-nameservice/hbase
```

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

## Post-migration Verification

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

## Upgrading from an Earlier CDH4 Release

- **Important:**

If you are using Cloudera Manager to manage CDH, do not use the instructions on this page. Follow the directions under Upgrading to the Latest Version of CDH4 in a Cloudera Managed Deployment in the [Cloudera Manager Installation Guide](#) (or the directions in the Installation Guide for the Cloudera Manager 4.x version you are using) to upgrade to the latest version of CDH4 in a Cloudera-managed deployment.

- **Important:**

- **Use the right instructions:** the following instructions describe how to upgrade to the latest CDH4 release from an earlier CDH4 release. If you are upgrading from a CDH3 release, use the instructions under [Upgrading from CDH3 to CDH4](#) instead.
- **MapReduce v1 (MRv1) and MapReduce v2 (YARN):** this page covers upgrade for MapReduce v1 (MRv1) and MapReduce v2 (YARN). MRv1 and YARN share common configuration files, so it is safe to *configure* both of them so long as you *run* only one set of daemons at any one time. Cloudera **does not** support running MRv1 and YARN daemons on the same nodes at the same time; it will degrade performance and may result in an unstable cluster deployment. Before deciding to deploy YARN, make sure you read the discussion on the [CDH4 Installation](#) page under [MapReduce 2.0 \(YARN\)](#).

- **Note: Running Services**

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

The following sections provide the information and instructions you need:

- [Before You Begin](#)
- [Upgrading from an Earlier CDH4 Release to the Latest Version](#)

## Before You Begin

- **Note:**

- Before upgrading, be sure to read about the latest [Incompatible Changes](#) and [Known Issues and Work Arounds in CDH4](#) in the [CDH4 Release Notes](#).
- If you are upgrading a cluster that is part of a production system, be sure to plan ahead. As with any operational work, be sure to reserve a maintenance window with enough extra time allotted in case of complications. The Hadoop upgrade process is well understood, but it is best to be cautious. For production clusters, Cloudera recommends allocating up to a full day maintenance window to perform the upgrade, depending on the number of hosts, the amount of experience you have with Hadoop and Linux, and the particular hardware you are using.

## Upgrading from an Earlier CDH4 Release

- 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 from CDH4, and then restore your data.
- If you have a multi-machine cluster, read the following sections to learn how to upgrade your cluster to CDH4.

- **Note: If you are using a high-availability (HA) configuration on CDH4 Beta 1:**

You must unconfigure HA before you proceed. See [Upgrading an HA Configuration to the Latest Release](#).

## Upgrading to the Latest Version of CDH4

Use the instructions that follow to upgrade to the latest version of CDH4.

### Step 1: Prepare the cluster for the upgrade.

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

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

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

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

- **Important:**

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

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

- **Note:**

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

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

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

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

```
$ cd /mnt/hadoop/hdfs/name
# tar -cvf /root/nn_backup_data.tar .
./
./current/
```

```
./current/fsimage
./current/fstime
./current/VERSION
./current/edits
./image/
./image/fsimage
```

■ **Warning:**

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

## Step 2: Download the CDH4 package on each of the hosts in your cluster.

**Before you begin:** Check whether you have the CDH4 "1-click" repository installed.

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

```
rpm -q cdh4-repository
```

If you are upgrading from CDH4 Beta 1 or later, you should see:

```
cdh4-repository-1-0
```

In this case, skip to [Step 3](#). If instead you see:

```
package cdh4-repository is not installed
```

proceed with [this step](#).

- On Ubuntu and Debian systems:

```
dpkg -l | grep cdh4-repository
```

If the repository is installed, skip to [Step 3](#); otherwise proceed with [this step](#).

If the CDH4 "1-click" repository is not already installed on each host in the cluster, follow the instructions below for that host's operating system:

[Instructions for Red Hat-compatible systems](#)

[Instructions for SLES systems](#)

[Instructions for Ubuntu and Debian systems](#)

### On Red Hat-compatible systems:

1. 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/Oracle 5	<a href="#">Red Hat/CentOS/Oracle 5 link</a>
Red Hat/CentOS 6 (32-bit)	<a href="#">Red Hat/CentOS 6 link (32-bit)</a>
Red Hat/CentOS/Oracle 6 (64-bit)	<a href="#">Red Hat/CentOS/Oracle 6 link (64-bit)</a>

2. Install the RPM.

## Upgrading from an Earlier CDH4 Release

For Red Hat/CentOS/Oracle 5:

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

For Red Hat/CentOS 6 (32-bit):

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.i386.rpm
```

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

```
$ sudo yum --nogpgcheck localinstall cloudera-cdh-4-0.x86_64.rpm
```

- **Note:**

For instructions on how to add a CDH4 yum repository or build your own CDH4 yum repository, see [Installing CDH4 On Red Hat-compatible systems](#).

On SLES systems:

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 cloudera-cdh-4-0.x86_64.rpm
```

- **Note:**

For instructions on how to add a repository or build your own repository, see [Installing CDH4 on SLES Systems](#).

Now update your system package index by running:

```
$ sudo zypper refresh
```

On Ubuntu and Debian systems:

1. Click one of the following: [this link for a Squeeze system](#), or [this link for a Lucid system](#), or [this link for a Precise 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 cdh4-repository_1.0_all.deb
```

- **Note:**

For instructions on how to add a repository or build your own repository, see [Installing CDH4 on Ubuntu Systems](#).

### Step 3: Upgrade the packages on the appropriate hosts.

Upgrade [MRv1](#), [YARN](#), or both, depending on what you intend to use.

- **Note:**

- Remember that you can install and configure both MRv1 and YARN, but you should not run them both on the same set of nodes at the same time.
- If you are using [HA for the NameNode](#), do not install `hadoop-hdfs-secondarynamenode`. Follow the upgrade instructions under [Upgrading an HDFS HA Configuration to the Latest Release](#).

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

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

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/5/x86_64/cdh/RPM-GPG-KEY-cloudera
```

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

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For all SLES systems:**

```
$ sudo rpm --import
http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/RPM-GPG-KEY-cloudera
```

- **For Ubuntu Lucid systems:**

```
$ curl -s
http://archive.cloudera.com/cdh4/ubuntu/lucid/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Ubuntu Precise systems:**

```
$ curl -s
http://archive.cloudera.com/cdh4/ubuntu/precise/amd64/cdh/archive.key
| sudo apt-key add -
```

- **For Debian Squeeze systems:**

```
$ curl -s
http://archive.cloudera.com/cdh4/debian/squeeze/amd64/cdh/archive.key
| sudo apt-key add -
```

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

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

1. Install and deploy ZooKeeper:

- **Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

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



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

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

1. Install and deploy ZooKeeper:

■ **Important:**

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

Follow instructions under [ZooKeeper Installation](#).

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

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

## Upgrading from an Earlier CDH4 Release

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

- **Note:**

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

### Step 4: Upgrade the HDFS Metadata (Beta 1 or earlier)

- **Note: If you are already running CDH4 Beta 2 or later, skip this step**

You do not need to upgrade the HDFS metadata. Proceed with [starting HDFS](#).

To upgrade to the latest version of CDH4 **from CDH4 Beta 1 or any earlier version of CDH**, you must now upgrade the HDFS metadata on the NameNode.

- **Note: Before you start**

If you are using a [high-availability \(HA\) configuration](#) on CDH4 Beta 1, you must unconfigure HA before you can upgrade HDFS. See [Upgrading an HA Configuration to the Latest Release](#).

1. To upgrade the HDFS metadata, run the following command on the NameNode:

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

■ **Note:**

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

You can watch the progress of the upgrade by running:

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

Look for a line that confirms the upgrade is complete, such as:

```
/var/lib/hadoop-hdfs/cache/hadoop/dfs/<name> is complete
```

2. Start up the DataNodes:

On each DataNode:

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

3. Wait for NameNode to exit safe mode, and then start the Secondary NameNode (if used) and complete the cluster upgrade.

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

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

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

## Step 5: Start HDFS (Beta 2 or later)

■ **Note:**

If you are upgrading the HDFS metadata, skip this step, and start the NameNode, DataNodes, and Secondary NameNode (if used) individually as described in the [previous step](#).

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

## Step 5a: Verify that /tmp Exists and Has the Right Permissions

■ **Important:**

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

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

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

Note:

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

Step 6: Start MapReduce (MRv1) or YARN

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

For MRv1	For YARN
<a href="#">Start MRv1</a>	<a href="#">Start YARN and the MapReduce JobHistory Server</a>
<a href="#">Verify basic cluster operation</a>	<a href="#">Verify basic cluster operation</a>

Step 6a: Start MapReduce (MRv1)

Important:

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

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

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

On the JobTracker system:

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

Verify that the JobTracker and TaskTracker started properly.

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

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

Verify basic cluster operation for MRv1.

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

Before you proceed, you make sure the `HADOOP_HOME` environment variable is unset:

```
$ unset HADOOP_HOME
```

Note:

To submit MapReduce jobs using MRv1 in CDH4 Beta 1, you needed either to set the `HADOOP_HOME` environment variable or run a launcher script.  
  
This is no longer true in later CDH4 releases; the `HADOOP_HOME` has been now fully deprecated and it is good practice to unset it.

■ **Note:**

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

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

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

Do the following steps as the user joe.

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

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

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

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

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

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

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

5. List the output files.

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

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

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

You have now confirmed your cluster is successfully running CDH4.

■ **Important:**

If you have client hosts, make sure you also update them to CDH4, and upgrade the [components](#) running on those clients as well.

## Upgrading from an Earlier CDH4 Release

### Step 6b: Start MapReduce with YARN

- **Important:**

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

Before deciding to deploy YARN, make sure you read the discussion under [New Features](#).

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

- **Note:** For more information see [Deploying MapReduce v2 \(YARN\) on a Cluster](#).

Create a history directory and set permissions; for example:

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

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

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

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

Verify the directory structure, ownership, and permissions:

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

You should see:

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

To start YARN, start the ResourceManager and NodeManager services:

- **Note:**

Make sure you always start ResourceManager before starting NodeManager services.

On the ResourceManager system:

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

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

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

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

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

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

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

### Verify basic cluster operation for YARN.

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

■ **Note:**

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

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

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

Do the following steps as the user `joe`.

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

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

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

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

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

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

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

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

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

6. List the output files:

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

## Upgrading from an Earlier CDH4 Release

7. Read the results in the output file:

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

You have now confirmed your cluster is successfully running CDH4.

■ **Important:**

If you have client hosts, make sure you also update them to CDH4, and upgrade the [components](#) running on those clients as well.

### Step 7: Set the Sticky Bit

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

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

### Step 8: Upgrade Components to CDH4

■ **Note:**

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

To upgrade or add CDH components, see the following sections:

- Flume. For more information, see "Upgrading Flume in CDH4" under "Flume Installation" in this guide.
- Sqoop. For more information, see "Upgrading Sqoop to CDH4" under "Sqoop Installation" in this guide.
- Sqoop 2. For more information, see "Sqoop 2 Installation" in this guide.
- HCatalog. For more information, see "Installing and Using HCatalog" in this guide.
- Hue. For more information, see "Upgrading Hue in CDH4" under "Hue Installation" in this guide.
- Pig. For more information, see "Upgrading Pig to CDH4" under "Pig Installation" in this guide.
- Hive. For more information, see "Upgrading Hive to CDH4" under "Hive Installation" in this guide.
- HBase. For more information, see "Upgrading HBase to CDH4" under "HBase Installation" in this guide.
- ZooKeeper. For more information, see "Upgrading ZooKeeper to CDH4" under "ZooKeeper Installation" in this guide.
- Oozie. For more information, see "Upgrading Oozie to CDH4" under "Oozie Installation" in this guide.
- Whirr. For more information, see "Upgrading Whirr to CDH4" under "Whirr Installation" in this guide.
- Snappy. For more information, see "Upgrading Snappy to CDH4" under "Snappy Installation" in this guide.
- Mahout. For more information, see "Upgrading Mahout to CDH4" under "Mahout Installation" in this guide.



## Step 9: Apply Configuration File Changes

- **Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by `dpkg`](#).

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

## Step 10: Finalize the HDFS Metadata Upgrade (Beta 1 or earlier)

- **Note:** Skip this step if you are upgrading from CDH4 Beta 2 or later.

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

1. Make sure you are satisfied that the CDH4 upgrade has succeeded and everything is running smoothly. This could take a matter of days, or even weeks.

- **Warning:**

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

- **Note:**

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

2. Finalize the HDFS metadata upgrade: use one of the following commands, depending on whether Kerberos is enabled (see [Configuring Hadoop Security in CDH4](#)).

- If Kerberos is enabled:

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

- If Kerberos is not enabled:

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

- **Note:**

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



## Configuring Ports for CDH4

The CDH4 components, and third parties such as Kerberos, use the ports listed in the tables that follow. Before you deploy CDH4, make sure these ports are open on each system. If you cannot follow the recommendations exactly, you must completely disable `iptables`.

- [Ports Used by Components of CDH4](#) on page 75
- [Ports Used by Third-Party Components](#) on page 79

### Ports Used by Components of CDH4

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

## Configuring Ports for CDH4

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	JournalNode		8480	TCP	Internal		
Hadoop MRv1	JobTracker		8021	TCP	External	mapred.job.tracker	
	JobTracker		50030	TCP	External	mapred.job.tracker.http.address	
	JobTracker	Thrift Plugin	9290	TCP	Internal	jobtracker.thrift.address	Required by Hue and Cloudera Manager Activity Monitor
	TaskTracker		50060	TCP	External	mapred.task.tracker.http.address	
	TaskTracker		0	TCP	Localhost	mapred.task.tracker.report.address	Communicating with child (umbilical)
Hadoop YARN	ResourceManager		8032	TCP		yarn.resourcemanager.address	
	ResourceManager		8030	TCP		yarn.resourcemanager.scheduler.address	
	ResourceManager		8031	TCP		yarn.resourcemanager.resource-tracker.address	
	ResourceManager		8033	TCP		yarn.resourcemanager.admin.address	
	ResourceManager		8088	TCP		yarn.resourcemanager.webapp.address	
	NodeManager		8040	TCP		yarn.nodemanager.localizer.address	
	NodeManager		8042	TCP		yarn.nodemanager.webapp.address	
	NodeManager		8041	TCP		yarn.nodemanager.address	
	MapReduce JobHistory Server		10020	TCP		mapreduce.jobhistory.address	
	MapReduce JobHistory Server		19888	TCP		mapreduce.jobhistory.webapp.address	
HBase	Master		60000	TCP	External	hbase.master.port	IPC

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	Master		60010	TCP	External	<code>hbase.master.info.port</code>	HTTP
	RegionServer		60020	TCP	External	<code>hbase.regionserver.port</code>	IPC
	RegionServer		60030	TCP	External	<code>hbase.regionserver.info.port</code>	HTTP
	HQuorumPeer		2181	TCP		<code>hbase.zookeeper.property.clientPort</code>	HBase-managed ZK mode
	HQuorumPeer		2888	TCP		<code>hbase.zookeeper.peerport</code>	HBase-managed ZK mode
	HQuorumPeer		3888	TCP		<code>hbase.zookeeper.leaderport</code>	HBase-managed ZK mode
	REST	REST Service	8080	TCP	External	<code>hbase.rest.port</code>	
	REST UI		8085	TCP	External		
	ThriftServer	Thrift Server	9090	TCP	External	Pass <code>-p &lt;port&gt;</code> on CLI	
	ThriftServer		9095	TCP	External		
		Avro server	9090	TCP	External	Pass <code>--port &lt;port&gt;</code> on CLI	
Hive	Metastore		9083	TCP	External		
	HiveServer		10000	TCP	External		
	HiveServer2		10000	TCP	External	<code>hive.server2.thrift.port</code>	
Sqoop	Metastore		16000	TCP	External	<code>sqoop.metastore.server.port</code>	
Sqoop 2	Sqoop 2 server		12000	TCP	External		
Sqoop 2	Sqoop 2		12001	TCP	External		Admin port
ZooKeeper	Server (with CDH4 and/or Cloudera Manager 4)		2181	TCP	External	<code>clientPort</code>	Client port
	Server (with CDH4 only)		2888	TCP	Internal	X in <code>server.N=host:X:Y</code>	Peer

## Configuring Ports for CDH4

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
	Server (with CDH4 only)		3888	TCP	Internal	X in server.N =host:X:Y	Peer
	Server (with CDH4 and Cloudera Manager 4)		3181	TCP	Internal	X in server.N =host:X:Y	Peer
	Server (with CDH4 and Cloudera Manager 4)		4181	TCP	Internal	X in server.N =host:X:Y	Peer
	ZooKeeper FailoverController (ZKFC)		8019	TCP	Internal		Used for HA
	ZooKeeper JMX port		9010	TCP	Internal		ZooKeeper will also use another randomly selected port for RMI. In order for Cloudera Manager to monitor ZooKeeper, you must open up all ports when the connection originates from the Cloudera Manager server.
Hue	Server		8888	TCP	External		
	Beeswax Server		8002		Internal		
	Beeswax Metastore		8003		Internal		
Oozie	Oozie Server		11000	TCP	External	OOZIE_HTTP_PORT in oozie-env.sh	HTTP
	Oozie Server		11001	TCP	localhost	OOZIE_ADMIN_PORT in oozie-env.sh	Shutdown port
HttpFS	HttpFS		14000	TCP			
	HttpFS		14001	TCP			

## Ports Used by Third-Party Components

Component	Service	Qualifier	Port	Protocol	Access Requirement	Configuration	Comment
Ganglia	ganglia-gmond		8649	UDP/TCP	Internal		
	ganglia-web		80	TCP	External	Via Apache httpd	
Kerberos	KRB5 KDC Server	Secure	88	UDP/TCP	External	kdc_ports and kdc_tcp_ports in either the [kdcdefaults] or [realms] sections of kdc.conf	By default only UDP
	KRB5 Admin Server	Secure	749	TCP	Internal	kadmind_port in the [realms] section of kdc.conf	





## Deploying CDH4 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 CDH4 in pseudo-distributed mode (and to install it if you have not already done so) follow the instructions in the [CDH4 Quick Start Guide](#).



## Deploying CDH4 on a Cluster

To deploy CDH4 on a cluster, do the following:

1. [Configure Network Hosts](#)
2. [Configure HDFS](#)
3. [Deploy MapReduce v1 \(MRv1\) or YARN with MapReduce v2 \(YARN\)](#)

See [Hadoop Users in CDH4](#) for a list of special users created by CDH4 during installation.

■ **Note:** Do the tasks in this section after installing the latest version of CDH; see [CDH4 Installation](#).

## Configuring Network Names

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

■ **Important:**  
CDH requires IPv4. IPv6 is not supported.

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

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

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

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

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

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

3. Make sure the `/etc/sysconfig/network` file on each system contains the hostname you have just set (or verified) for that system, for example `myhost-1`.
4. Check that this system is consistently identified to the network:
  - a. Run `uname -a` and check that the hostname matches the output of the `hostname` command.
  - b. Run `/sbin/ifconfig` and note the value of `inet addr` in the `eth0` entry, for example:

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

```

## Deploying CDH4 on a Cluster

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

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

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

## Deploying HDFS on a Cluster

■ **Important:**

For instructions for configuring High Availability (HA) for the NameNode, see the [CDH4 High Availability Guide](#).

Proceed as follows to deploy HDFS on a cluster. Do this for all clusters, whether you are deploying MRv1 or YARN:

1. [Copy the Hadoop configuration](#)
2. [Customize configuration files](#)
3. [Configure Local Storage Directories](#)
4. [Configure DataNodes to tolerate local storage directory failure](#)
5. [Format the NameNode](#)
6. [Configure a remote NameNode storage directory](#)
7. [Configure the Secondary NameNode \(if used\)](#)
8. [Optionally enable Trash](#)
9. [Optionally configure DataNode storage balancing](#)
10. [Optionally enable WebHDFS](#)
11. [Optionally configure LZ0](#)
12. [Deploy MRv1 or YARN and start services](#)

## Copying the Hadoop Configuration

To customize the Hadoop configuration:

1. Copy the default configuration to your custom directory:

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

You can call this configuration anything you like; in this example, it's called `my_cluster`.

- Set alternatives to point to your custom directory, as follows.

To manually set the configuration on Red Hat-compatible systems:

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

To manually set the configuration on Ubuntu and SLES systems:

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

For more information on alternatives, see the `update-alternatives(8)` man page on Ubuntu and SLES systems or the `alternatives(8)` man page On Red Hat-compatible systems.

■ **Important:**

When performing the configuration tasks in this section, and when you go on to deploy MRv1 or YARN, edit the configuration files in your custom directory (for example `/etc/hadoop/conf.my_cluster`). Do not create your custom configuration in the default directory `/etc/hadoop/conf.dist`.

## Customizing Configuration Files

The following tables show the most important properties that you must configure for your cluster.

■ **Note:**

For information on other important configuration properties, and the configuration files, see the [Apache Cluster Setup](#) page.

Property	Configuration File	Description
<code>fs.defaultFS</code>	<code>conf/core-site.xml</code>	Note: <code>fs.default.name</code> is deprecated. Specifies the NameNode and the default file system, in the form <code>hdfs://&lt;namenode host&gt;:&lt;namenode port&gt;/</code> . The default value is <code>file:///</code> . The default file system is used to resolve relative paths; for example, if <code>fs.default.name</code> or <code>fs.defaultFS</code> is set to <code>hdfs://mynamenode/</code> , the relative URI <code>/mydir/myfile</code> resolves to <code>hdfs://mynamenode/mydir/myfile</code> . Note: for the cluster to function correctly, the <code>&lt;namenode&gt;</code> part of the string <b>must</b> be the hostname (for example <code>mynamenode</code> ) not the IP address.
<code>dfs.permissions.superusergroup</code>	<code>conf/hdfs-site.xml</code>	Specifies the UNIX group containing users that will be treated as superusers by HDFS. You can stick

Property	Configuration File	Description
		<p>with the value of <code>hadoop</code> or pick your own group depending on the security policies at your site.</p> <p>The <code>hdfs</code>, <code>yarn</code>, and <code>mapred</code> users belong to the <code>hadoop</code> group. To give users root privileges in HDFS, create a UNIX group with the same name as this group (or change the value of the configuration to correspond to an existing UNIX group) and add them to the group. The <code>impala</code> user also belongs to the <code>hdfs</code> group.</p>

Sample Configuration

**core-site.xml:**

```
<property>
  <name>fs.defaultFS</name>
  <value>hdfs://namenode-host.company.com/</value>
</property>
```

**hdfs-site.xml:**

```
<property>
  <name>dfs.permissions.superusergroup</name>
  <value>hadoop</value>
</property>
```

Configuring Local Storage Directories

You need to specify, create, and assign the correct permissions to the local directories where you want the HDFS daemons to store data. You specify the directories by configuring the following two properties in the `hdfs-site.xml` file.

Property	Configuration File Location	Description
<code>dfs.name.dir</code> or <code>dfs.namenode.name.dir</code>	<code>hdfs-site.xml</code> on the NameNode	This property specifies the URIs of the directories where the NameNode stores its metadata and edit logs. Cloudera recommends that you specify at least two directories. One of these should be located on an NFS mount point, unless you will be using a <a href="#">High Availability (HA) configuration</a> .

Property	Configuration File Location	Description
		<p><b>Note:</b> Cludera recommends you specify a directory which will not exist if the NFS mount point is not present. This will prevent the NameNode from filling up the local filesystem when NFS fails to mount at all.</p>
dfs.data.dir or dfs.datanode.data.dir	hdfs-site.xml on each DataNode	This property specifies the URIs of the directories where the DataNode stores blocks. Cludera recommends that you configure the disks on the DataNode in a JBOD configuration, mounted at /data/1/ through /data/N, and configure dfs.data.dir or dfs.datanode.data.dir to specify file:///data/1/dfs/dn through file:///data/N/dfs/dn/.

**Note:**

dfs.data.dir and dfs.name.dir are deprecated; you should use dfs.datanode.data.dir and dfs.namenode.name.dir instead, though dfs.data.dir and dfs.name.dir will still work.

Sample configuration:

**hdfs-site.xml on the NameNode:**

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

**hdfs-site.xml on each DataNode:**

```
<property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///data/1/dfs/dn,/data/2/dfs/dn,/data/3/dfs/dn</value>
</property>
```

After specifying these directories as shown above, you must create the directories and assign the correct file permissions to them on each node in your cluster.

In the following instructions, local path examples are used to represent Hadoop parameters. Change the path examples to match your configuration.

**Local directories:**

- The dfs.name.dir or dfs.namenode.name.dir parameter is represented by the /data/1/dfs/nn and /nfsmount/dfs/nn path examples.
- The dfs.data.dir or dfs.datanode.data.dir parameter is represented by the /data/1/dfs/dn, /data/2/dfs/dn, /data/3/dfs/dn, and /data/4/dfs/dn path examples.

**To configure local storage directories for use by HDFS:**

## Deploying CDH4 on a Cluster

1. On a NameNode host: create the `dfs.name.dir` or `dfs.namenode.name.dir` local directories:

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

- **Important:**

If you are using [High Availability \(HA\)](#), you should **not** configure these directories on an NFS mount; configure them on local storage.

2. On all DataNode hosts: create the `dfs.data.dir` or `dfs.datanode.data.dir` local directories:

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

3. Configure the owner of the `dfs.name.dir` or `dfs.namenode.name.dir` directory, and of the `dfs.data.dir` or `dfs.datanode.data.dir` directory, to be the `hdfs` user:

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

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

Directory	Owner	Permissions (see Footnote 1)
<code>dfs.name.dir</code> or <code>dfs.namenode.name.dir</code>	<code>hdfs:hdfs</code>	<code>drwx-----</code>
<code>dfs.data.dir</code> or <code>dfs.datanode.data.dir</code>	<code>hdfs:hdfs</code>	<code>drwx-----</code>

**Footnote: 1** The Hadoop daemons automatically set the correct permissions for you on `dfs.data.dir` or `dfs.datanode.data.dir`. But in the case of `dfs.name.dir` or `dfs.namenode.name.dir`, permissions are currently incorrectly set to the file-system default, usually `drwxr-xr-x` (755). Use the `chmod` command to reset permissions for these `dfs.name.dir` or `dfs.namenode.name.dir` directories to `drwx-----` (700); for example:

```
$sudo chmod 700 /data/1/dfs/nn /nfsmount/dfs/nn
```

or

```
$sudo chmod go-rx /data/1/dfs/nn /nfsmount/dfs/nn
```

- **Note:**

If you specified nonexistent directories for the `dfs.data.dir` or `dfs.datanode.data.dir` property in the `conf/hdfs-site.xml` file, CDH4 will shut down. (In previous releases, CDH3 silently ignored nonexistent directories for `dfs.data.dir`.)

## Configuring DataNodes to Tolerate Local Storage Directory Failure

By default, the failure of a single `dfs.data.dir` or `dfs.datanode.data.dir` will cause the HDFS DataNode process to shut down, which results in the NameNode scheduling additional replicas for each block that is present on the DataNode. This causes needless replications of blocks that reside on disks that have not failed.

To prevent this, you can configure DataNodes to tolerate the failure of `dfs.data.dir` or `dfs.datanode.data.dir` directories; use the `dfs.datanode.failed.volumes.tolerated` parameter in `hdfs-site.xml`. For example, if the value for this parameter is 3, the DataNode will only shut down after four or more data directories have



failed. This value is respected on DataNode startup; in this example the DataNode will start up as long as no more than three directories have failed.

■ **Note:**

It is important that `dfs.datanode.failed.volumes.tolerated` not be configured to tolerate too many directory failures, as the DataNode will perform poorly if it has few functioning data directories.

## Formatting the NameNode

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

■ **Important:**

- Make sure you format the NameNode as user `hdfs`.
- If you are re-formatting the NameNode, keep in mind that this invalidates the DataNode storage locations, so you should remove the data under those locations after the NameNode is formatted.

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

■ **Note:**

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) or `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

You'll get a confirmation prompt; for example:

```
Re-format filesystem in /data/namedir ? (Y or N)
```

Respond with an **upper-case** `y`; if you use lower case, the process will abort.

## Configuring a Remote NameNode Storage Directory

You should configure the NameNode to write to multiple storage directories, including one remote NFS mount. To keep NameNode processes from hanging when the NFS server is unavailable, configure the NFS mount as a `soft` mount (so that I/O requests that time out fail rather than hang), and set other options as follows:

```
tcp,soft,intr,timeo=10,retrans=10
```

These options configure a soft mount over TCP; transactions will be retried ten times (`retrans=10`) at 1-second intervals (`timeo=10`) before being deemed to have failed.

**Example:**

```
mount -t nfs -o tcp,soft,intr,timeo=10,retrans=10, <server>:<export> <mount_point>
```

where `<server>` is the remote host, `<export>` is the exported file system, and `<mount_point>` is the local mount point.

■ **Note:**

Cloudera recommends similar settings for shared HA mounts, as in the example that follows.

### Example for HA:

```
mount -t nfs -o tcp,soft,intr,timeo=50,retrans=12, <server>:<export> <mount_point>
```

Note that in the HA case `timeo` should be set to 50 (five seconds), rather than 10 (1 second), and `retrans` should be set to 12, giving an overall timeout of 60 seconds.

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

### Configuring Remote Directory Recovery

You can enable the `dfs.namenode.name.dir.restore` option so that the NameNode will attempt to recover a previously failed NameNode storage directory on the next checkpoint. This is useful for restoring a remote storage directory mount that has failed because of a network outage or intermittent NFS failure.

## Configuring the Secondary NameNode

### ■ Important:

The Secondary NameNode does not provide failover or High Availability (HA). If you intend to configure [HA for the NameNode](#), skip this section: do not install or configure the Secondary Name Node (the Standby NameNode performs checkpointing). After completing the software configuration for your [chosen HA method](#), follow the installation instructions under [HDFS High Availability Initial Deployment](#).

In non-HA deployments, configure a Secondary NameNode that will periodically merge the EditLog with the FSImage, creating a new FSImage which incorporates the changes which were in the EditLog. This reduces the amount of disk space consumed by the EditLog on the NameNode, and also reduces the restart time for the Primary NameNode.

A standard Hadoop cluster (not a Hadoop Federation or HA configuration), can have only one Primary NameNode plus one Secondary NameNode. On production systems, the Secondary NameNode should run on a different machine from the Primary NameNode to improve scalability (because the Secondary NameNode does not compete with the NameNode for memory and other resources to create the system snapshot) and durability (because the copy of the metadata is on a separate machine that is available if the NameNode hardware fails).

### Configuring the Secondary NameNode on a Separate Machine

To configure the Secondary NameNode on a separate machine from the NameNode, proceed as follows.

1. Add the name of the machine that will run the Secondary NameNode to the `conf/masters` file.
2. Add the following property to the `hdfs-site.xml` file:

```
<property>
  <name>dfs.namenode.http-address</name>
  <value><namenode.host.address>:50070</value>
  <description>
    The address and the base port on which the dfs NameNode Web UI will listen.
  </description>
</property>
```

■ **Note:**

- `dfs.http.address` is deprecated; use `dfs.namenode.http-address`.
- In most cases, you should set `dfs.namenode.http-address` to a routable IP address with port 50070. However, in some cases such as Amazon EC2, when the NameNode should bind to multiple local addresses, you may want to set `dfs.namenode.http-address` to `0.0.0.0:50070` *on the NameNode machine only*, and set it to a real, routable address on the Secondary NameNode machine. The different addresses are needed in this case because HDFS uses `dfs.namenode.http-address` for two different purposes: it defines both the address the NameNode binds to, and the address the Secondary NameNode connects to for checkpointing. Using `0.0.0.0` on the NameNode allows the NameNode to bind to all its local addresses, while using the externally-routable address on the the Secondary NameNode provides the Secondary NameNode with a real address to connect to.

For more information, see [Multi-host SecondaryNameNode Configuration](#).

### More about the Secondary NameNode

- The NameNode stores the HDFS metadata information in RAM to speed up interactive lookups and modifications of the metadata.
- For reliability, this information is flushed to disk periodically. To ensure that these writes are not a speed bottleneck, only the list of modifications is written to disk, not a full snapshot of the current filesystem. The list of modifications is appended to a file called `edits`.
- Over time, the `edits` log file can grow quite large and consume large amounts of disk space.
- When the NameNode is restarted, it takes the HDFS system state from the `fsimage` file, then applies the contents of the `edits` log to construct an accurate system state that can be loaded into the NameNode's RAM. If you restart a large cluster that has run for a long period with no Secondary NameNode, the `edits` log may be quite large, and so it can take some time to reconstruct the system state to be loaded into RAM.

When the Secondary NameNode is configured, it periodically (once an hour, by default) constructs a **checkpoint** by compacting the information in the `edits` log and merging it with the most recent `fsimage` file; it then clears the `edits` log. So, when the NameNode restarts, it can use the latest checkpoint and apply the contents of the smaller `edits` log.

### Secondary NameNode Parameters

The behavior of the Secondary NameNode is controlled by the following parameters in `{hdfs-site.xml}`.

- `dfs.namenode.checkpoint.check.period`
- `dfs.namenode.checkpoint.txns`
- `dfs.namenode.checkpoint.dir`
- `dfs.namenode.checkpoint.edits.dir`
- `dfs.namenode.num.checkpoints.retained`

See <http://archive.cloudera.com/cdh4/cdh/4/hadoop/hadoop-project-dist/hadoop-hdfs/hdfs-default.xml> for details.

## Enabling Trash

■ **Important:**

The trash feature is disabled by default. Cloudera recommends that you enable it on all production clusters.

The Hadoop trash feature helps prevent accidental deletion of files and directories. If trash is enabled and a file or directory is deleted using the Hadoop shell, the file is moved to the `.Trash` directory in the user's home

directory instead of being deleted. Deleted files are initially moved to the `Current` sub-directory of the `.Trash` directory, and their original path is preserved. If trash checkpointing is enabled, the `Current` directory is periodically renamed using a timestamp. Files in `.Trash` are permanently removed after a user-configurable time delay. Files and directories in the trash can be restored simply by moving them to a location outside the `.Trash` directory.

■ **Note:**

The trash feature works by default only for files and directories deleted using the Hadoop shell. Files or directories deleted programmatically using other interfaces (WebHDFS or the Java APIs, for example) are not moved to trash, even if trash is enabled, unless the program has implemented a call to the trash functionality. (Hue, for example, implements trash as of CDH 4.4.)

Users can bypass trash when deleting files using the shell by specifying the `-skipTrash` option to the `hadoop fs -rm -r` command. This can be useful when it is necessary to delete files that are too large for the user's quota.

Trash is configured with the following properties in the `core-site.xml` file:

CDH4 Parameter	Value	Description
<code>fs.trash.interval</code>	<i>minutes or 0</i>	<p>The number of minutes after which a trash checkpoint directory is deleted. This option can be configured both on the server and the client; in releases prior to CDH4.1 this option could be configured only on the client.</p> <ul style="list-style-type: none"> <li>▪ If trash is enabled in the server configuration, then the value configured on the server is used and the client configuration is ignored.</li> <li>▪ If trash is disabled in the server configuration, then the client side configuration is checked.</li> <li>▪ If the value of this property is zero (the default), then the trash feature is disabled.</li> </ul>
<code>fs.trash.checkpoint.interval</code>	<i>minutes or 0</i>	<p>The number of minutes between trash checkpoints. Every time the checkpointer runs on the NameNode, it creates a new checkpoint of the "Current" directory and removes checkpoints older than <code>fs.trash.interval</code> minutes. This value should be smaller than or equal to <code>fs.trash.interval</code>. This option is configured on the server. If configured to zero (the default), then the value is set to the value of <code>fs.trash.interval</code>.</p>

For example, to enable trash so that files deleted using the Hadoop shell are not deleted for 24 hours, set the value of the `fs.trash.interval` property in the server's `core-site.xml` file to a value of 1440.

■ **Note:**

The period during which a file remains in the trash starts when the file is moved to the trash, not when the file is last modified.

## Configuring Storage-Balancing for the DataNodes

You can configure HDFS to distribute writes on each DataNode in a manner that balances out available storage among that DataNode's disk volumes.

By default a DataNode writes new block replicas to disk volumes solely on a round-robin basis. As of CDH4.3, you can configure a volume-choosing policy that causes the DataNode to take into account how much space is available on each volume when deciding where to place a new replica.

You can configure

- how much DataNode volumes are allowed to differ in terms of bytes of free disk space before they are considered imbalanced, *and*
- what percentage of new block allocations will be sent to volumes with more available disk space than others.

To configure storage balancing, set the following properties in `hdfs-site.xml`.

- **Note:** Keep in mind that if usage is markedly imbalanced among a given DataNode's storage volumes when you enable storage balancing, throughput on that DataNode will be affected initially, as writes are disproportionately directed to the under-utilized volumes.

Property	Value
<code>dfs.datanode.fsdataset.volume.choosing.policy</code>	<code>org.apache.hadoop</code>
<code>dfs.datanode.available-space-volume-choosing-policy.balanced-space-threshold</code>	10737418240 (default)
<code>dfs.datanode.available-space-volume-choosing-policy.balanced-space-preference-fraction</code>	0.75 (default)

## Enabling WebHDFS

If you want to use WebHDFS, you must first enable it.

**To enable WebHDFS:**

Set the following property in `hdfs-site.xml`:

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

- **Note:**  
If you want to use WebHDFS in a secure cluster, you must set additional properties to configure secure WebHDFS. For instructions, see [Configure secure WebHDFS](#).

## Configuring LZO

If you have [installed LZO](#), configure it as follows.

**To configure LZO:**

Set the following property in `core-site.xml`.

■ **Note:**

If you copy and paste the *value* string, make sure you remove the line-breaks and carriage returns, which are included below because of page-width constraints.

```
<property>
  <name>io.compression.codecs</name>

  <value>org.apache.hadoop.io.compress.DefaultCodec,org.apache.hadoop.io.compress.GzipCodec,
  org.apache.hadoop.io.compress.BZip2Codec,com.hadoop.compression.lzo.LzoCodec,
  com.hadoop.compression.lzo.LzopCodec,org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

For more information about LZ0, see [Using LZ0 Compression](#).

## Deploy MRv1 or YARN

The next step is to deploy MRv1 or YARN and start HDFS services. See

- [Deploying MapReduce v1 \(MRv1\) on a Cluster](#) or
- [Deploying MapReduce v2 \(YARN\) on a Cluster](#)

## Deploying MapReduce v1 (MRv1) on a Cluster

This section describes configuration and startup tasks for MRv1 clusters only.

■ **Important:**

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

- If you have [installed CDH4 from packages](#), MRv1 is the default deployment. Follow the instructions on this page. (If you have [installed YARN](#) and want to deploy it instead of MRv1, read the discussion under [New Features](#) and then follow [these instructions](#) instead of the ones below.)
- If you have installed CDH4 from tarballs, the default deployment is YARN.

Do these tasks after you have [configured HDFS](#):

1. [Configure properties for MRv1 clusters](#)
2. [Configure local storage directories for use by MRv1 daemons](#)
3. [Configure a health check script for DataNode processes](#)
4. [Configure JobTracker Recovery](#)
5. [Deploy the configuration to the entire cluster](#)
6. [Start HDFS](#)
7. [Create the HDFS /tmp directory](#)
8. [Create MapReduce /var directories](#)
9. [Verify the HDFS File Structure](#)
10. [Create and configure the mapred.system.dir directory in HDFS](#)
11. [Start MapReduce](#)
12. [Create a Home Directory for each MapReduce User](#)
13. [Configure the Hadoop daemons to start at boot time](#)

■ **Note: Running Services**

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

## Step 1: Configuring Properties for MRv1 Clusters

■ **Note:**

Edit these files in the custom directory you created when you [copied the Hadoop configuration](#). When you have finished, you will push this configuration to all the nodes in the cluster; see [Step 4](#).

For instructions on configuring a highly available JobTracker, see [Configuring High Availability for the JobTracker \(MRv1\)](#); you need to configure `mapred.job.tracker` differently in that case, and you must not use the port number.

Property	Configuration File	Description
<code>mapred.job.tracker</code>	<code>conf/mapred-site.xml</code>	If you plan to run your cluster with MRv1 daemons you need to specify the hostname and (optionally) port of the JobTracker's RPC server, in the form <code>&lt;host&gt;:&lt;port&gt;</code> . See <a href="#">Configuring Ports for CDH4</a> for the default port. If the value is set to <code>local</code> , the default, the JobTracker runs on demand when you run a MapReduce job; do not try to start the JobTracker yourself in this case. Note: if you specify the host (rather than using <code>local</code> ) this <b>must</b> be the hostname (for example <code>mynamenode</code> ) not the IP address.

Sample configuration:

**mapred-site.xml:**

```
<property>
  <name>mapred.job.tracker</name>
  <value>jobtracker-host.company.com:8021</value>
</property>
```

## Step 2: Configure Local Storage Directories for Use by MRv1 Daemons

For MRv1, you need to configure an additional property in the `mapred-site.xml` file.

Property	Configuration File Location	Description
mapred.local.dir	mapred-site.xml on each TaskTracker	This property specifies the directories where the TaskTracker will store temporary data and intermediate map output files while running MapReduce jobs. Cloudera recommends that this property specifies a directory on each of the JBOD mount points; for example, /data/1/mapred/local through /data/N/mapred/local.

Sample configuration:

**mapred-site.xml on each TaskTracker:**

```
<property>
  <name>mapred.local.dir</name>
  <value>/data/1/mapred/local,/data/2/mapred/local,/data/3/mapred/local</value>
</property>
```

After specifying these directories in the mapred-site.xml file, you must create the directories and assign the correct file permissions to them on each node in your cluster.

**To configure local storage directories for use by MapReduce:**

In the following instructions, local path examples are used to represent Hadoop parameters. The mapred.local.dir parameter is represented by the /data/1/mapred/local, /data/2/mapred/local, /data/3/mapred/local, and /data/4/mapred/local path examples. Change the path examples to match your configuration.

- 1. Create the mapred.local.dir local directories:

```
$ sudo mkdir -p /data/1/mapred/local /data/2/mapred/local /data/3/mapred/local
/data/4/mapred/local
```

- 2. Configure the owner of the mapred.local.dir directory to be the mapred user:

```
$ sudo chown -R mapred:hadoop /data/1/mapred/local /data/2/mapred/local
/data/3/mapred/local /data/4/mapred/local
```

The correct owner and permissions of these local directories are:

Owner	Permissions	
mapred:hadoop	drwxr-xr-x	

Step 3: Configure a Health Check Script for DataNode Processes

In earlier releases, the failure of a single mapred.local.dir caused the MapReduce TaskTracker process to shut down, resulting in the machine not being available to execute tasks. In CDH4, the TaskTracker process will continue to execute tasks as long as it has a single functioning mapred.local.dir available. No configuration change is necessary to enable this behavior.

Because a TaskTracker that has few functioning local directories will not perform well, Cloudera recommends configuring a health script that checks if the DataNode process is running (if configured as described under [Configuring DataNodes to Tolerate Local Storage Directory Failure](#), the DataNode will shut down after the



configured number of directory failures). Here is an example health script that exits if the DataNode process is not running:

```
#!/bin/bash
if ! jps | grep -q DataNode ; then
  echo ERROR: datanode not up
fi
```

In practice, the `dfs.data.dir` and `mapred.local.dir` are often configured on the same set of disks, so a disk failure will result in the failure of both a `dfs.data.dir` and `mapred.local.dir`.

See the section titled "Configuring the Node Health Check Script" in [the Apache cluster setup documentation](#) for further details.

## Step 4: Configure JobTracker Recovery

JobTracker recovery means that jobs that are running when JobTracker fails (for example, because of a system crash or hardware failure) are re-run when the JobTracker is restarted. Any jobs that were running at the time of the failure will be re-run from the beginning automatically.

A recovered job will have the following properties:

- It will have the same job ID as when it was submitted.
- It will run under the same user as the original job.
- It will write to the same output directory as the original job, overwriting any previous output.
- It will show as RUNNING on the JobTracker web page after you restart the JobTracker.

## Enabling JobTracker Recovery

By default JobTracker recovery is off, but you can enable it by setting the property `mapreduce.jobtracker.restart.recover` to true in `mapred-site.xml`.

## Step 5: Deploy your Custom Configuration to your Entire Cluster

To deploy your configuration to your entire cluster:

1. Push your custom directory (for example `/etc/hadoop/conf.my_cluster`) to each node in your cluster; for example:

```
$ sudo scp -r /etc/hadoop/conf.my_cluster
myuser@myCDHnode-<n>.mycompany.com: /etc/hadoop/conf.my_cluster
```

2. Manually set alternatives on each node to point to that directory, as follows.

**To manually set the configuration on Red Hat-compatible systems:**

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

**To manually set the configuration on Ubuntu and SLES systems:**

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

For more information on alternatives, see the `update-alternatives(8)` man page on Ubuntu and SLES systems or the `alternatives(8)` man page On Red Hat-compatible systems.

## Deploying CDH4 on a Cluster

### Step 6: Start HDFS on Every Node in the Cluster

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

### Step 7: Create the HDFS /tmp Directory

- **Important:**

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

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

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

- **Note:**

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

### Step 8: Create MapReduce /var directories

```
sudo -u hdfs hadoop fs -mkdir -p /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chmod 1777 /var/lib/hadoop-hdfs/cache/mapred/mapred/staging
sudo -u hdfs hadoop fs -chown -R mapred /var/lib/hadoop-hdfs/cache/mapred
```

### Step 9: Verify the HDFS File Structure

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

You should see:

```
drwxrwxrwt - hdfs supergroup          0 2012-04-19 15:14 /tmp
drwxr-xr-x - hdfs supergroup           0 2012-04-19 15:16 /var
drwxr-xr-x - hdfs supergroup           0 2012-04-19 15:16 /var/lib
drwxr-xr-x - hdfs supergroup           0 2012-04-19 15:16 /var/lib/hadoop-hdfs
drwxr-xr-x - hdfs supergroup          0 2012-04-19 15:16 /var/lib/hadoop-hdfs/cache
drwxr-xr-x - mapred supergroup          0 2012-04-19 15:19
/var/lib/hadoop-hdfs/cache/mapred
drwxr-xr-x - mapred supergroup          0 2012-04-19 15:29
/var/lib/hadoop-hdfs/cache/mapred/mapred
drwxrwxrwt - mapred supergroup          0 2012-04-19 15:33
/var/lib/hadoop-hdfs/cache/mapred/mapred/staging
```

### Step 10: Create and Configure the `mapred.system.dir` Directory in HDFS

After you start HDFS and create `/tmp`, but before you start the JobTracker (see the [next step](#)), you must also create the HDFS directory specified by the `mapred.system.dir` parameter (by default `${hadoop.tmp.dir}/mapred/system` and configure it to be owned by the `mapred` user.

To create the directory in its default location:

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

■ **Important:**

If you create the `mapred.system.dir` directory in a different location, specify that path in the `conf/mapred-site.xml` file.

When starting up, MapReduce sets the permissions for the `mapred.system.dir` directory to `drwx-----`, assuming the user `mapred` owns that directory.

## Step 11: Start MapReduce

To start MapReduce, start the TaskTracker and JobTracker services

On each TaskTracker system:

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

On the JobTracker system:

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

## Step 12: Create a Home Directory for each MapReduce User

Create a home directory for each MapReduce user. It is best to do this on the NameNode; for example:

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

where `<user>` is the Linux username of each user.

Alternatively, you can log in as each Linux user (or write a script to do so) and create the home directory as follows:

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

## Configure the Hadoop Daemons to Start at Boot Time

See [Configuring the Hadoop Daemons to Start at Boot Time](#).

## Deploying MapReduce v2 (YARN) on a Cluster

This section describes configuration tasks for YARN clusters only, and is specifically tailored for administrators who have [installed YARN from packages](#).

Do these tasks after you have configured HDFS:

1. [Configure properties for YARN clusters](#)
2. [Configure YARN daemons](#)
3. [Configure the History Server](#)
4. [Configure the Staging Directory](#)
5. [Deploy your custom configuration to your entire cluster](#)
6. [Start HDFS](#)

## Deploying CDH4 on a Cluster

7. [Create the HDFS /tmp directory](#)
8. [Create the History Directory and Set Permissions](#)
9. [Create Log Directories](#)
10. [Verify the HDFS File Structure](#)
11. [Start YARN and the MapReduce JobHistory Server](#)
12. [Create a home directory for each MapReduce user](#)
13. [Set HADOOP\\_MAPRED\\_HOME](#)
14. [Configure the Hadoop daemons to start at boot time](#)

### ■ **Note: Running Services**

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

## About MapReduce v2 (YARN)

MapReduce has undergone a complete overhaul and CDH4 now includes MapReduce 2.0 (MRv2). The fundamental idea of MRv2's YARN architecture is to split up the two primary responsibilities of the JobTracker — resource management and job scheduling/monitoring — into separate daemons: a global ResourceManager (RM) and per-application ApplicationMasters (AM). With MRv2, the ResourceManager (RM) and per-node NodeManagers (NM), form the data-computation framework. The ResourceManager service effectively replaces the functions of the JobTracker, and NodeManagers run on slave nodes instead of TaskTracker daemons. The per-application ApplicationMaster is, in effect, a framework specific library and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the tasks. For details of the new architecture, see [Apache Hadoop NextGen MapReduce \(YARN\)](#).

- **Note:** Cloudera does not consider the current upstream MRv2 release stable yet, and it could potentially change in non-backwards-compatible ways. Cloudera recommends that you use MRv1 unless you have particular reasons for using MRv2, which should not be considered production-ready.

For more information about the two implementations (MRv1 and MRv2) see the discussion under Apache Hadoop MapReduce in the "What's New in Beta 1" section of [New Features](#).

See also [Selecting Appropriate JAR files for your MRv1 and YARN Jobs](#).

### ■ **Important:**

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

- If you have [installed YARN from packages](#) and want to deploy it, read the discussion under [New Features in CDH4](#) and then follow the instructions below. (To deploy MRv1 instead, see [Deploying MapReduce v1 \(MRv1\) on a Cluster](#).)
- If you have installed CDH4 from tarballs, the default deployment is YARN. Keep in mind that the instructions on this page are tailored for a deployment following installation from packages.

## Step 1: Configure Properties for YARN Clusters

**Note:**

Edit these files in the custom directory you created when you [copied the Hadoop configuration](#). When you have finished, you will push this configuration to all the nodes in the cluster; see [Step 4](#).

Property	Configuration File	Description
mapreduce.framework.name	conf/mapred-site.xml	If you plan on running YARN, you must set this property to the value of yarn.

Sample Configuration:

**mapred-site.xml:**

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

## Step 2: Configure YARN daemons

If you have decided to run YARN, you must configure the following services: ResourceManager (on a dedicated host) and NodeManager (on every host where you plan to run MapReduce v2 jobs).

The following table shows the most important properties that you must configure for your cluster in yarn-site.xml

Property	Recommended value	Description
yarn.nodemanager.aux-services	mapreduce.shuffle	Shuffle service that needs to be set for Map Reduce applications.
yarn.nodemanager.aux-services.mapreduce.shuffle.class	org.apache.hadoop.mapred.ShuffleHandler	The exact name of the class for shuffle service
yarn.resourcemanager.address	resourcemanager.company.com:8032	The address of the applications manager interface in the RM.
yarn.resourcemanager.scheduler.address	resourcemanager.company.com:8030	The address of the scheduler interface.
yarn.resourcemanager.resource-tracker.address	resourcemanager.company.com:8031	The address of the resource tracker interface.
yarn.resourcemanager.admin.address	resourcemanager.company.com:8033	The address of the RM admin interface.
yarn.resourcemanager.webapp.address	resourcemanager.company.com:8088	The address of the RM web application.
yarn.application.classpath	\$HADOOP_CONF_DIR, \$HADOOP_COMMON_HOME/*, \$HADOOP_COMMON_HOME/lib/*, \$HADOOP_HDFS_HOME/*, \$HADOOP_HDFS_HOME/lib/*, \$HADOOP_MAPRED_HOME/*,	Classpath for typical applications.

Property	Recommended value	Description
	\$HADOOP_MAPRED_HOME/lib/*, \$YARN_HOME/*, \$YARN_HOME/lib/* \$HADOOP_YARN_HOME/*, \$HADOOP_YARN_HOME/lib/*	

Next, you need to specify, create, and assign the correct permissions to the local directories where you want the YARN daemons to store data.

You specify the directories by configuring the following two properties in the `yarn-site.xml` file on all cluster nodes:

Property	Description
<code>yarn.nodemanager.local-dirs</code>	Specifies the URIs of the directories where the NodeManager stores its localized files. All of the files required for running a particular YARN application will be put here for the duration of the application run. Cloudera recommends that this property specify a directory on each of the JBOD mount points; for example, <code>file:///data/1/yarn/local</code> through <code>file:///data/N/yarn/local</code> .
<code>yarn.nodemanager.log-dirs</code>	Specifies the URIs of the directories where the NodeManager stores container log files. Cloudera recommends that this property specify a directory on each of the JBOD mount points; for example, <code>file:///data/1/yarn/logs</code> through <code>file:///data/N/yarn/logs</code> .
<code>yarn.nodemanager.remote-app-log-dir</code>	Specifies the URI of the directory where logs are aggregated. Set the value to <code>hdfs://var/log/hadoop-yarn/apps</code> . See also <a href="#">Step 9</a> .

Here is an example configuration:

**yarn-site.xml:**

```
<property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>resourcemanager.company.com:8031</value>
</property>
<property>
  <name>yarn.resourcemanager.address</name>
  <value>resourcemanager.company.com:8032</value>
</property>
<property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>resourcemanager.company.com:8030</value>
</property>
<property>
  <name>yarn.resourcemanager.admin.address</name>
  <value>resourcemanager.company.com:8033</value>
</property>
<property>
  <name>yarn.resourcemanager.webapp.address</name>
  <value>resourcemanager.company.com:8088</value>
</property>
<property>
  <description>Classpath for typical applications.</description>
```

```

<name>yarn.application.classpath</name>
<value>
    $HADOOP_CONF_DIR,
    $HADOOP_COMMON_HOME/*,$HADOOP_COMMON_HOME/lib/*,
    $HADOOP_HDFS_HOME/*,$HADOOP_HDFS_HOME/lib/*,
    $HADOOP_MAPRED_HOME/*,$HADOOP_MAPRED_HOME/lib/*,
    $YARN_HOME/*,$YARN_HOME/lib/*
</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce.shuffle</value>
</property>
<property>
    <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
    <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
<property>
    <name>yarn.nodemanager.local-dirs</name>
<value>file:///data/1/yarn/local,file:///data/2/yarn/local,file:///data/3/yarn/local</value>
</property>
<property>
    <name>yarn.nodemanager.log-dirs</name>
<value>file:///data/1/yarn/logs,file:///data/2/yarn/logs,file:///data/3/yarn/logs</value>
</property>
<property>
    <description>Where to aggregate logs</description>
    <name>yarn.nodemanager.remote-app-log-dir</name>
    <value>hdfs://var/log/hadoop-yarn/apps</value>
</property>

```

After specifying these directories in the `yarn-site.xml` file, you must create the directories and assign the correct file permissions to them on each node in your cluster.

In the following instructions, local path examples are used to represent Hadoop parameters. Change the path examples to match your configuration.

#### To configure local storage directories for use by YARN:

1. Create the `yarn.nodemanager.local-dirs` local directories:

```
$ sudo mkdir -p /data/1/yarn/local /data/2/yarn/local /data/3/yarn/local
/data/4/yarn/local
```

2. Create the `yarn.nodemanager.log-dirs` local directories:

```
$ sudo mkdir -p /data/1/yarn/logs /data/2/yarn/logs /data/3/yarn/logs
/data/4/yarn/logs
```

3. Configure the owner of the `yarn.nodemanager.local-dirs` directory to be the `yarn` user:

```
$ sudo chown -R yarn:yarn /data/1/yarn/local /data/2/yarn/local /data/3/yarn/local
/data/4/yarn/local
```

4. Configure the owner of the `yarn.nodemanager.log-dirs` directory to be the `yarn` user:

```
$ sudo chown -R yarn:yarn /data/1/yarn/logs /data/2/yarn/logs /data/3/yarn/logs
/data/4/yarn/logs
```

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

Directory	Owner	Permissions
yarn.nodemanager.local-dirs	yarn:yarn	drwxr-xr-x
yarn.nodemanager.log-dirs	yarn:yarn	drwxr-xr-x

### Step 3: Configure the History Server

If you have decided to run YARN on your cluster instead of MRv1, you should also run the MapReduce JobHistory Server. The following table shows the most important properties that you must configure in `mapred-site.xml`:

Property	Recommended value	Description
mapreduce.jobhistory.address	historyserver.company.com:10020	The address of the JobHistory Server host:port
mapreduce.jobhistory.webapp.address	historyserver.company.com:19888	The address of the JobHistory Server web application host:port

In addition, make sure proxying is enabled for the `mapred` user; configure the following properties in `core-site.xml`:

Property	Recommended value	Description
hadoop.proxyuser.mapred.groups	*	Allows the <code>mapreduser</code> to move files belonging to users in these groups
hadoop.proxyuser.mapred.hosts	*	Allows the <code>mapreduser</code> to move files belonging on these hosts

### Step 4: Configure the Staging Directory

YARN requires a staging directory for temporary files created by running jobs. By default it creates `/tmp/hadoop-yarn/staging` with restrictive permissions that may prevent your users from running jobs. To forestall this, you should configure and create the staging directory yourself; in the example that follows we use `/user`:

1. Configure `yarn.app.mapreduce.am.staging-dir` in `mapred-site.xml`:

```
<property>
  <name>yarn.app.mapreduce.am.staging-dir</name>
  <value>/user</value>
</property>
```

2. Once HDFS is up and running, you will create this directory and a `history` subdirectory under it (see [Step 8](#)).

Alternatively, you can do the following:

1. Configure `mapreduce.jobhistory.intermediate-done-dir` and `mapreduce.jobhistory.done-dir` in `yarn-site.xml`.
2. Create these two directories.
3. Set permissions on `mapreduce.jobhistory.intermediate-done-dir` to 1777.
4. Set permissions on `mapreduce.jobhistory.done-dir` to 750.

If you configure `mapreduce.jobhistory.intermediate-done-dir` and `mapreduce.jobhistory.done-dir` as above, you can skip [Step 8](#).



## Step 5: Deploy your custom Configuration to your Entire Cluster

To deploy your configuration to your entire cluster:

1. Push your custom directory (for example `/etc/hadoop/conf.my_cluster`) to each node in your cluster; for example:

```
$ sudo scp -r /etc/hadoop/conf.my_cluster
myuser@myCDHnode-<n>.mycompany.com: /etc/hadoop/conf.my_cluster
```

2. Manually set `alternatives` on each node to point to that directory, as follows.

**To manually set the configuration on Red Hat-compatible systems:**

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

**To manually set the configuration on Ubuntu and SLES systems:**

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

For more information on `alternatives`, see the `update-alternatives(8)` man page on Ubuntu and SLES systems or the `alternatives(8)` man page On Red Hat-compatible systems.

## Step 6: Start HDFS on Every Node in the Cluster

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

## Step 7: Create the HDFS /tmp Directory

### ■ Important:

If you do not create `/tmp` properly, with the right permissions as shown below, you may have problems with CDH components later. Specifically, if you don't create `/tmp` yourself, another process may create it automatically with restrictive permissions that will prevent your other applications from using it.

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

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

### ■ Note:

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

### Step 8: Create the history Directory and Set Permissions and Owner

This is a subdirectory of the staging directory you configured in [Step 4](#). In this example we're using `/user/history`. Create it and set permissions as follows:

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

### Step 9: Create Log Directories

- **Note:**  
See also [Step 2](#).

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

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

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

### Step 10: Verify the HDFS File Structure:

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

You should see:

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

### Step 11: Start YARN and the MapReduce JobHistory Server

To start YARN, start the ResourceManager and NodeManager services:

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

On the ResourceManager system:

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

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

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

To start the MapReduce JobHistory Server

On the MapReduce JobHistory Server system:

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

## Step 12: Create a Home Directory for each MapReduce User

Create a home directory for each MapReduce user. It is best to do this on the NameNode; for example:

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

where <user> is the Linux username of each user.

Alternatively, you can log in as each Linux user (or write a script to do so) and create the home directory as follows:

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

## Step 13: Set HADOOP\_MAPRED\_HOME

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

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

## Step 14: Configure the Hadoop Daemons to Start at Boot Time

See [Configuring the Hadoop Daemons to Start at Boot Time](#).

# Configuring the Daemons to Start in an MRv1 Cluster

### ■ Important:

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

To start the Hadoop daemons at boot time and on restarts, enable their `init` scripts on the systems on which the services will run, using the `chkconfig` tool. See [Configuring init to Start Core Hadoop System Services](#).

Non-core services can also be started at boot time; after you install the non-core components, see [Configuring init to Start Non-core Hadoop System Services](#) for instructions.

## Tips and Guidelines

### Selecting Appropriate JAR files for your MRv1 and YARN Jobs

Each implementation of the CDH4 MapReduce framework (MRv1 and YARN) consists of the artifacts (JAR files) that provide MapReduce functionality as well as auxiliary utility artifacts that are used during the course of the MapReduce job. When you submit a job either explicitly (using the Hadoop launcher script) or implicitly (via Java implementations) it is extremely important that you make sure that you reference utility artifacts that come with the same version of MapReduce implementation that is running on your cluster. The following table summarizes the names and location of these artifacts:

Name	MRv1 location	YARN location
streaming	/usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.0.0-mr1-cdh<version>.jar	/usr/lib/hadoop-mapreduce/hadoop-streaming.jar
rumen	N/A	/usr/lib/hadoop-mapreduce/hadoop-rumen.jar
hadoop examples	/usr/lib/hadoop-0.20-mapreduce/hadoop-examples.jar	/usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar
distcp v1	/usr/lib/hadoop-0.20-mapreduce/hadoop-tools.jar	/usr/lib/hadoop-mapreduce/hadoop-extras.jar
distcp v2	N/A	/usr/lib/hadoop-mapreduce/hadoop-distcp.jar
hadoop archives	/usr/lib/hadoop-0.20-mapreduce/hadoop-tools.jar	/usr/lib/hadoop-mapreduce/hadoop-archives.jar

## Improving Performance

This section provides solutions to some performance problems, and describes configuration best practices.

### Important:

If you are running CDH over 10Gbps Ethernet, improperly set network configuration or improperly applied NIC firmware or drivers can noticeably degrade performance. Work with your network engineers and hardware vendors to make sure that you have the proper NIC firmware, drivers, and configurations in place and that your network performs properly. Cloudera recognizes that network setup and upgrade are challenging problems, and will make best efforts to share any helpful experiences.

## Disabling Transparent Hugepage Compaction

Most Linux platforms supported by CDH4 include a feature called **transparent hugepage compaction** which interacts poorly with Hadoop workloads and can seriously degrade performance.

**Symptom:** `top` and other system monitoring tools show a large percentage of the CPU usage classified as "system CPU". If system CPU usage is 30% or more of the total CPU usage, your system may be experiencing this issue.

**What to do:**

### Note:

- Red Hat/CentOS: `/sys/kernel/mm/redhat_transparent_hugepage/defrag`
- Ubuntu/Debian, OEL, SLES: `/sys/kernel/mm/transparent_hugepage/defrag`

- To see whether transparent hugepage compaction is enabled, run the following command and check the output:

```
$ cat defrag_file_pathname
```

- `[always]` never means that transparent hugepage compaction is enabled.
- `always [never]` means that transparent hugepage compaction is disabled.

- To disable transparent hugepage compaction, add the following command to `/etc/rc.local`:

```
echo never > defrag_file_pathname
```

You can also disable transparent hugepage compaction interactively (but remember this will not survive a reboot).

**To disable transparent hugepage compaction temporarily as root:**

```
# echo 'never' > defrag_file_pathname
```

**To disable transparent hugepage compaction temporarily using sudo:**

```
$ sudo sh -c "echo 'never' > defrag_file_pathname"
```

### Setting the vm.swappiness Linux Kernel Parameter

`vm.swappiness` is a Linux Kernel Parameter that controls how aggressively memory pages are swapped to disk. It can be set to a value between 0-100; the higher the value, the more aggressive the kernel is in seeking out inactive memory pages and swapping them to disk.

You can see what value `vm.swappiness` is currently set to by looking at `/proc/sys/vm`; for example:

```
cat /proc/sys/vm/swappiness
```

On most systems, it is set to 60 by default. This is not suitable for Hadoop clusters nodes, because it can cause processes to get swapped out even when there is free memory available. This can affect stability and performance, and may cause problems such as lengthy garbage collection pauses for important system daemons. Cloudera recommends that you set this parameter to 0; for example:

```
# sysctl -w vm.swappiness=0
```

### Improving Performance in Shuffle Handler and IFile Reader

As of CDH4.1, the MapReduce shuffle handler and IFile reader use native Linux calls (`posix_fadvise(2)` and `sync_data_range`) on Linux systems with Hadoop native libraries installed. The subsections that follow provide details.

#### Shuffle Handler

You can improve MapReduce Shuffle Handler Performance by enabling shuffle readahead. This causes the TaskTracker or Node Manager to pre-fetch map output before sending it over the socket to the reducer.

- To enable this feature for YARN, set the `mapreduce.shuffle.manage.os.cache` property to `true` (default). To further tune performance, adjust the value of the `mapreduce.shuffle.readahead.bytes` property. The default value is 4MB.
- To enable this feature for MRv1, set the `mapred.tasktracker.shuffle.fadvise` property to `true` (default). To further tune performance, adjust the value of the `mapred.tasktracker.shuffle.readahead.bytes` property. The default value is 4MB.

#### IFile Reader

Enabling IFile readahead increases the performance of merge operations. To enable this feature for either MRv1 or YARN, set the `mapreduce.ifile.readahead` property to `true` (default). To further tune the performance, adjust the value of the `mapreduce.ifile.readahead.bytes` property. The default value is 4MB.

### Best Practices for MapReduce Configuration

The configuration settings described below can reduce inherent latencies in MapReduce execution. You set these values in `mapred-site.xml`.

**Send a heartbeat as soon as a task finishes**

## Deploying CDH4 on a Cluster

Set the `mapreduce.tasktracker.outofband.heartbeat` property to `true` to let the TaskTracker send an out-of-band heartbeat on task completion to reduce latency; the default value is `false`:

```
<property>
  <name>mapreduce.tasktracker.outofband.heartbeat</name>
  <value>true</value>
</property>
```

### Reduce the interval for JobClient status reports on single node systems

The `jobclient.progress.monitor.poll.interval` property defines the interval (in milliseconds) at which JobClient reports status to the console and checks for job completion. The default value is 1000 milliseconds; you may want to set this to a lower value to make tests run faster on a single-node cluster. Adjusting this value on a large production cluster may lead to unwanted client-server traffic.

```
<property>
  <name>jobclient.progress.monitor.poll.interval</name>
  <value>10</value>
</property>
```

### Tune the JobTracker heartbeat interval

Tuning the minimum interval for the TaskTracker-to-JobTracker heartbeat to a smaller value may improve MapReduce performance on small clusters.

```
<property>
  <name>mapreduce.jobtracker.heartbeat.interval.min</name>
  <value>10</value>
</property>
```

### Start MapReduce JVMs immediately

The `mapred.reduce.slowstart.completed.maps` property specifies the proportion of Map tasks in a job that must be completed before any Reduce tasks are scheduled. For small jobs that require fast turnaround, setting this value to 0 can improve performance; larger values (as high as 50%) may be appropriate for larger jobs.

```
<property>
  <name>mapred.reduce.slowstart.completed.maps</name>
  <value>0</value>
</property>
```

## Best practices for HDFS Configuration

This section indicates changes you may want to make in `hdfs-site.xml`.

### Improve Performance for Local Reads

#### Note:

Also known as **short-circuit local reads**, this capability is particularly useful for HBase and Cloudera Impala™. It improves the performance of node-local reads by providing a fast path that is enabled in this case. It requires `libhadoop.so` (the [Hadoop Native Library](#)) to be accessible to both the server and the client.

`libhadoop.so` is not available if you have installed CDH from a tarball. You must install from an `.rpm`, `.deb`, or `parcel` in order to use short-circuit local reads.

Configure the following properties in `hdfs-site.xml` as shown:

```
<property>
  <name>dfs.client.read.shortcircuit</name>
  <value>true</value>
```

```

</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.size</name>
  <value>1000</value>
</property>

<property>
  <name>dfs.client.read.shortcircuit.streams.cache.expiry.ms</name>
  <value>10000</value>
</property>

<property>
  <name>dfs.domain.socket.path</name>
  <value>/var/run/hadoop-hdfs/dn._PORT</value>
</property>

```

■ **Note:**

The text `_PORT` appears just as shown; you do not need to substitute a number.

If `/var/run/hadoop-hdfs/` is group-writable, make sure its group is `root`.

To check that short-circuit reads are working in the case of HBase, you can run the Linux command `lsot` on a Region Server. If any files are open in the directories identified by `dfs.datanode.data.dirs`, short-circuit reads are working. (See [Configuring Local Storage Directories](#) on page 86 for more information about `dfs.datanode.data.dirs` (or `dfs.data.dirs`).

### Tips and Best Practices for Jobs

This section describes changes you can make at the job level.

#### Use the Distributed Cache to Transfer the Job JAR

Use the distributed cache to transfer the job JAR rather than using the `JobConf(Class)` constructor and the `JobConf.setJar()` and `JobConf.setJarByClass()` method.

To add JARs to the classpath, use `-libjars <jar1>,<jar2>`, which will copy the local JAR files to HDFS and then use the distributed cache mechanism to make sure they are available on the task nodes and are added to the task classpath.

The advantage of this over `JobConf.setJar` is that if the JAR is on a task node it won't need to be copied again if a second task from the same job runs on that node, though it will still need to be copied from the launch machine to HDFS.

■ **Note:**

`-libjars` works only if your MapReduce driver uses [ToolRunner](#). If it doesn't, you would need to use the DistributedCache APIs (Cloudera does not recommend this).

For more information, see item 1 in the blog post [How to Include Third-Party Libraries in Your MapReduce Job](#).

#### Changing the Logging Level on a Job (MRv1)

You can change the logging level for an individual job. You do this by setting the following properties in the job configuration (`JobConf`):

- `mapreduce.map.log.level`
- `mapreduce.reduce.log.level`

Valid values are `NONE`, `INFO`, `WARN`, `DEBUG`, `TRACE`, and `ALL`.

### Example:

```
JobConf conf = new JobConf();  
...  
conf.set("mapreduce.map.log.level", "DEBUG");  
conf.set("mapreduce.reduce.log.level", "TRACE");  
...
```

## Setting Quotas

As the system administrator, you can set quotas in HDFS for:

- The number of file and directory names used; *and*
- The amount of space used by given directories.

Points to note:

- The quotas for names and the quotas for space are independent of each other.
- File and directory creation fails if the creation would cause the quota to be exceeded.
- Allocation fails if the quota would prevent a full block from being written; keep this in mind if you are using a large block size.
- If you are using replication, remember that each replica of a block counts against the quota.

## Commands

To set space quotas on a directory:

```
dfsadmin -setSpaceQuota n directory
```

where *n* is a number of bytes and *directory* is the directory the quota applies to. You can specify multiple directories in a single command; *n* applies to each.

To remove space quotas from a directory:

```
dfsadmin -clrSpaceQuota directory
```

You can specify multiple directories in a single command.

To set name quotas on a directory:

```
dfsadmin -setQuota n directory
```

where *n* is the number of file and directory names in *directory*. You can specify multiple directories in a single command; *n* applies to each.

To remove name quotas from a directory:

```
dfsadmin -clrQuota directory
```

You can specify multiple directories in a single command.

## For More Information

For more information, see the [HDFS Quotas Guide](#).



# Flume Installation

Apache Flume is a distributed, reliable, and available system for efficiently collecting, aggregating and moving large amounts of log data from many different sources to a centralized data store.

**This guide is for Flume 1.x** (specifically the 1.3.0 release).

■ **Note:**

**If you are looking for Flume 0.9.x:**

For information on Flume 0.9.x, see the [Flume 0.9.x documentation](#). To install Flume 0.9.x instead of Flume 1.x, go to <http://archive.cloudera.com/cdh4-deprecated>.

You cannot install both Flume 0.9.x and Flume 1.x together on the same host.

The following sections provide more information and instructions:

- [Migrating from Flume 0.9x](#)
- [Packaging](#)
- [Installing a Tarball](#)
- [Installing Packages](#)
- [Configuring Flume](#)
- [Verifying the Installation](#)
- [Running Flume](#)
- [Files Installed by the Packages](#)
- [Supported Sources, Sinks, and Channels](#)
- [Using and On-disk Encrypted File Channel](#)
- [Apache Flume Documentation](#)

## Migrating to Flume 1.x from Flume 0.9.x

Flume 1.x is a significant departure from Flume 0.9.x in its implementation although many of the original concepts are the same. If you're already familiar with Flume, here are some significant points.

- You still have sources and sinks and they still do the same thing. They are now connected by channels.
- Channels are pluggable, and dictate durability. Flume 1.x ships with an in-memory channel for fast, but non-durable event delivery. There are also JDBC-based and file-based channels for durable event delivery.
- There are no more logical or physical nodes. All physical nodes are "agents," and agents can run zero or more sources and sinks.
- There is no longer a Master, and no ZooKeeper dependency. At this time, Flume runs with a simple file-based configuration system.
- Just about everything is a plugin — some end user facing, some for tool and system developers.
- Thrift and Avro legacy Flume sources are provided to enable sending events from Flume 0.9.4 to Flume 1.x.

An "upgrade" to Flume requires uninstalling Flume 0.9.x and then installing Flume 1.x. If you have already removed the old Flume as part of [Upgrading from CDH3 to CDH4](#) you can skip the first step.

### ■ **Note: Running Services**

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

### Step 1: Remove Flume 0.9.x from your cluster.

1. Stop the Flume Node processes on each node where they are running:

```
$ sudo service flume-node stop
```

2. Stop the Flume Master:

```
$ sudo service flume-master stop
```

3. Uninstall the old Flume components:

#### **On Red Hat-compatible systems:**

```
$ sudo yum remove flume
```

#### **On SLES systems:**

```
$ sudo zypper remove flume
```

#### **On Ubuntu systems:**

```
$ sudo apt-get remove flume
```

### Step 2. Install the new version of Flume

Follow the instructions in the rest of this document to install Flume 1.x.

## Flume Packaging

There are currently three packaging options available for installing Flume:

- Tarball (.tar.gz)
- RPM packages
- Debian packages

### Installing the Flume Tarball

The Flume tarball is a self-contained package containing everything needed to use Flume on a Unix-like system. To install Flume from the tarball, you unpack it in the appropriate directory.

■ **Note:**

The tarball does not come with any scripts suitable for running Flume as a service or daemon. This makes the tarball distribution appropriate for *ad hoc* installations and preliminary testing, but a more complete installation is provided by the binary RPM and Debian packages.

**To install the Flume tarball on Linux-based systems:**

1. Run the following commands, replacing the (component\_version) with the current version numbers for Flume and CDH.

```
$ cd /usr/local/lib
$ sudo tar -zxvf <path_to_flume-ng-(Flume_version)-cdh(CDH_version).tar.gz>
$ sudo mv flume-ng-(Flume_version)-cdh(CDH_version) flume-ng
```

For example,

```
$ cd /usr/local/lib
$ sudo tar -zxvf <path_to_flume-ng-1.3.0-cdh4.4.0.tar.gz>
$ sudo mv flume-ng-1.3.0-cdh4.4.0 flume-ng
```

2. To complete the configuration of a tarball installation, you must set your `PATH` variable to include the `bin/` subdirectory of the directory where you installed Flume. For example:

```
$ export PATH=/usr/local/lib/flume-ng/bin:$PATH
```

## Installing the Flume RPM or Debian Packages

Installing the Flume RPM and Debian packages is more convenient than installing the Flume tarball because the packages:

- Handle dependencies
- Provide for easy upgrades
- Automatically install resources to conventional locations
- Handle daemon startup and shutdown.

The Flume RPM and Debian packages consist of three packages:

- `flume-ng` — Everything you need to run Flume
- `flume-ng-agent` — Handles starting and stopping the Flume agent as a service
- `flume-ng-doc` — Flume documentation

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

■ **Important:**

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands to install Flume. For instructions, see [CDH4 Installation](#).

**To install Flume on Ubuntu and other Debian systems:**

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

**To install Flume On Red Hat-compatible systems:**

```
$ sudo yum install flume-ng
```

## Flume Installation

**To install Flume on SLES systems:**

```
$ sudo zypper install flume-ng
```

You may also want to enable automatic start-up on boot. To do this, install the Flume agent.

**To install the Flume agent so Flume starts automatically on boot on Ubuntu and other Debian systems:**

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

**To install the Flume agent so Flume starts automatically on boot on Red Hat-compatible systems:**

```
$ sudo yum install flume-ng-agent
```

**To install the Flume agent so Flume starts automatically on boot on SLES systems:**

```
$ sudo zypper install flume-ng-agent
```

To install the documentation:

**To install the documentation on Ubuntu and other Debian systems:**

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

**To install the documentation on Red Hat-compatible systems:**

```
$ sudo yum install flume-ng-doc
```

**To install the documentation on SLES systems:**

```
$ sudo zypper install flume-ng-doc
```

## Flume Configuration

Flume 1.x provides a template configuration file for `flume.conf` called `conf/flume-conf.properties.template` and a template for `flume-env.sh` called `conf/flume-env.sh.template`.

1. Copy the Flume template property file `conf/flume-conf.properties.template` to `conf/flume.conf`, then edit it as appropriate.

```
$ sudo cp conf/flume-conf.properties.template conf/flume.conf
```

This is where you define your sources, sinks, and channels, and the flow within an agent. By default, the properties file is configured to work out of the box using a sequence generator source, a logger sink, and a memory channel. For information on configuring agent flows in Flume 1.x, as well as more details about the [supported sources, sinks and channels](#), see the documents listed under [Viewing the Flume Documentation](#).

2. Optionally, copy the template `flume-env.sh` file `conf/flume-env.sh.template` to `conf/flume-env.sh`.

```
$ sudo cp conf/flume-env.sh.template conf/flume-env.sh
```

The `flume-ng` executable looks for a file named `flume-env.sh` in the `conf` directory, and sources it if it finds it. Some use cases for using `flume-env.sh` are to specify a bigger heap size for the flume agent, or to specify debugging or profiling options via `JAVA_OPTS` when developing your own custom Flume NG components such as sources and sinks. If you do not make any changes to this file, then you need not perform the copy as it is effectively empty by default.

## Verifying the Installation

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

```
$ flume-ng help
```

You should see something similar to this:

```
Usage: /usr/bin/flume-ng <command> [options]...

commands:
  help                display this help text
  agent               run a Flume agent
  avro-client          run an avro Flume client
  version              show Flume version info

global options:
  --conf,-c <conf>    use configs in <conf> directory
  --classpath,-C <cp> append to the classpath
  --dryrun,-d          do not actually start Flume, just print the command
  --Dproperty=value    sets a JDK system property value

agent options:
  --conf-file,-f <file> specify a config file (required)
  --name,-n <name>      the name of this agent (required)
  --help,-h             display help text

avro-client options:
  --rpcProps,-P <file>  RPC client properties file with server connection params
  --host,-H <host>      hostname to which events will be sent (required)
  --port,-p <port>      port of the avro source (required)
  --dirname <dir>       directory to stream to avro source
  --filename,-F <file>  text file to stream to avro source [default: std input]
  --headerFile,-R <file> headerFile containing headers as key/value pairs on each new
line
  --help,-h             display help text

Either --rpcProps or both --host and --port must be specified.

Note that if <conf> directory is specified, then it is always included first
in the classpath.
```

■ **Note:**

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

## Running Flume

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

```
$ sudo service flume-ng-agent <start | stop | restart>
```

You can also run the agent in the foreground directly by using the `flume-ng agent` command:

```
$ /usr/bin/flume-ng agent -c <config-dir> -f <config-file> -n <agent-name>
```

## Flume Installation

For example:

```
$ /usr/bin/flume-ng agent -c /etc/flume-ng/conf -f /etc/flume-ng/conf/flume.conf -n agent
```

## Files Installed by the Flume RPM and Debian Packages

Resource	Location	Notes
Config Directory	/etc/flume-ng/conf	
Config File	/etc/flume-ng/conf/flume.conf	This config will be picked-up by the flume agent startup script.
Template of User Customizable Config File	/etc/flume-ng/conf/flume-conf.properties.template	Contains a sample config. To use this config you should copy this file onto /etc/flume-ng/conf/flume.conf and then modify as appropriate
Template of User Customizable environment file	/etc/flume-ng/conf/flume-env.sh.template	If you want modify this file, copy it first and modify the copy
Daemon Log Directory	/var/log/flume-ng	Contains log files generated by flume agent
Default Flume Home	/usr/lib/flume-ng	Provided by RPMS and DEBS
Flume Agent startup script	/etc/init.d/flume-ng-agent	Provided by RPMS and DEBS
Recommended tar.gz Flume Home	/usr/local/lib/flume-ng	Recommended but installation dependent
Flume Wrapper Script	/usr/bin/flume-ng	Called by the Flume Agent startup script
Flume Agent configuration file	/etc/default/flume-ng-agent	Allows you to specify non-default values for the agent name and for the configuration file location

## Supported Sources, Sinks, and Channels

The following tables list the currently-supported sources, sinks, and channels. For more information, including information on developing custom components, see the documents listed under [Viewing the Flume Documentation](#).

### Sources

Type	Description	Implementation Class
avro	Avro Netty RPC event source. Listens on Avro port and receives events from external Avro streams.	AvroSource

Type	Description	Implementation Class
netcat	Netcat style TCP event source. Listens on a given port and turns each line of text into an event.	NetcatSource
seq	Monotonically incrementing sequence generator event source	SequenceGeneratorSource
exec	Execute a long-lived Unix process and read from stdout.	ExecSource
syslogtcp	Reads syslog data and generates flume events. Creates a new event for a string of characters separated by carriage return ( \n ).	SyslogTcpSource
syslogudp	Reads syslog data and generates flume events. Treats an entire message as a single event.	SyslogUDPSource
org.apache.flume.source.avroLegacy. AvroLegacySource	Allows the Flume 1.x agent to receive events from Flume 0.9.4 agents over avro rpc.	AvroLegacySource
org.apache.flume.source.thriftLegacy. ThriftLegacySource	Allows the Flume 1.x agent to receive events from Flume 0.9.4 agents over thrift rpc.	ThriftLegacySource
org.apache.flume.source.StressSource	Mainly for testing purposes. Not meant for production use. Serves as a continuous source of events where each event has the same payload.	StressSource
org.apache.flume.source.scribe. ScribeSource	Scribe event source. Listens on Scribe port and receives events from Scribe. <b>Note that in CDH4.2, Scribe Source is at an experimental stage of development and should not be considered production-ready.</b>	ScribeSource
multiport_syslogtcp	Multi-port capable version of the SyslogTcpSource.	MultiportSyslogTCPSource
spooldir	Used for ingesting data by placing files to be ingested into a "spooling" directory on disk.	SpoolDirectorySource
http	Accepts Flume events by HTTP POST and GET. GET should be used for experimentation only.	HTTPSource
org.apache.flume.source.jms.JMSSource	Reads messages from a JMS destination such as a queue or topic.	JMSSource
org.apache.flume.agent.embedded. EmbeddedSource	Used only by the Flume embedded agent. See <a href="#">Flume Developer Guide</a> for more details.	EmbeddedSource

Type	Description	Implementation Class
Other (custom)	You need to specify the fully-qualified name of the custom source, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's <code>lib</code> directory.	—

## Sinks

Type	Description	Implementation Class
null	/dev/null for Flume - blackhole all events received	NullSink
logger	Log events at INFO level via configured logging subsystem (log4j by default)	LoggerSink
avro	Sink that invokes a pre-defined Avro protocol method for all events it receives (when paired with an avro source, forms tiered collection)	AvroSink
hdfs	Writes all events received to HDFS (with support for rolling, bucketing, HDFS-200 append, and more)	HDFSEventSink
file_roll	Writes all events received to one or more files.	RollingFileSink
irc	Takes messages from attached channel and relays those to configured IRC destinations.	IRCSink
org.apache.flume.hbase.HBaseSink	A simple sink that reads events from a channel and writes them synchronously to HBase. The AsyncHBaseSink is recommended.	HBaseSink
org.apache.flume.sink.hbase.AsyncHBaseSink	A simple sink that reads events from a channel and writes them asynchronously to HBase. This is the recommended HBase sink, but note that it does not support Kerberos.	AsyncHBaseSink
org.apache.flume.sink.morphine.MorphineSolrSink	Extracts and transforms data from Flume events, and loads it into Apache Solr servers. See the section on MorphineSolrSink in the Flume User Guide listed under <a href="#">Viewing the Flume Documentation</a> on page 124.	MorphineSolrSink



Type	Description	Implementation Class
Other (custom)	You need to specify the fully-qualified name of the custom sink, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's <code>lib</code> directory.	—

## Channels

Type	Description	Implementation Class
memory	In-memory, fast, non-durable event transport	MemoryChannel
jdbc	JDBC-based, durable event transport (Derby-based)	JDBCChannel
file	File-based, durable event transport	FileChannel
Other (custom)	You need to specify the fully-qualified name of the custom channel, and provide that class (and its dependent code) in Flume's classpath. You can do this by creating a JAR file to hold the custom code, and placing the JAR in Flume's <code>lib</code> directory.	—

## Using an On-disk Encrypted File Channel

Beginning with CDH4.1, Flume supports on-disk encryption of data on the local disk. To implement this:

- Generate an encryption key to use for the Flume Encrypted File Channel
- Configure on-disk encryption by setting parameters in the `flume.conf` file

### Important:

Flume on-disk encryption operates with a maximum strength of 128-bit AES encryption unless the JCE unlimited encryption cryptography policy files are installed. Please see this Oracle document for information about enabling strong cryptography with JDK 1.6:

<http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html>

Consult your security organization for guidance on the acceptable strength of your encryption keys. Cloudera has tested with AES-128, AES-192, and AES-256.

## Generating Encryption Keys

Use the `keytool` program included in Oracle JDK 1.6 to create the AES encryption keys for use with Flume.

## Flume Installation

The command to generate a 128-bit key that uses the same password as the key store password is:

```
keytool -genseckey -alias key-1 -keyalg AES -keysize 128 -validity 9000 \  
-keystore test.keystore -storetype jceks \  
-storepass keyStorePassword
```

The command to generate a 128-bit key that uses a different password from that used by the key store is:

```
keytool -genseckey -alias key-0 -keypass keyPassword -keyalg AES \  
-keysize 128 -validity 9000 -keystore test.keystore \  
-storetype jceks -storepass keyStorePassword
```

The key store and password files can be stored anywhere on the file system; both files should have `flume` as the owner and `0600` permissions.

Please note that `-keysize` controls the strength of the AES encryption key, in bits; 128, 192, and 256 are the allowed values.

## Configuration

Flume on-disk encryption is enabled by setting parameters in the `/etc/flume-ng/conf/flume.conf` file.

### Basic Configuration

The first example is a basic configuration with an alias called `key-0` that uses the same password as the key store:

```
agent.channels.ch-0.type = file  
agent.channels.ch-0.capacity = 10000  
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING  
agent.channels.ch-0.encryption.activeKey = key-0  
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE  
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore  
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =  
/path/to/my.keystore.password  
agent.channels.ch-0.encryption.keyProvider.keys = key-0
```

In the next example, `key-0` uses its own password which may be different from the key store password:

```
agent.channels.ch-0.type = file  
agent.channels.ch-0.capacity = 10000  
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING  
agent.channels.ch-0.encryption.activeKey = key-0  
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE  
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore  
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =  
/path/to/my.keystore.password  
agent.channels.ch-0.encryption.keyProvider.keys = key-0  
agent.channels.ch-0.encryption.keyProvider.keys.key-0.passwordFile =  
/path/to/key-0.password
```

## Changing Encryption Keys Over Time

To modify the key, modify the configuration as shown below. This example shows how to change the configuration to use `key-1` instead of `key-0`:

```
agent.channels.ch-0.type = file  
agent.channels.ch-0.capacity = 10000  
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING  
agent.channels.ch-0.encryption.activeKey = key-1  
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE  
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore  
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
```

```
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0 key-1
```

The same scenario except that key-0 and key-1 have their own passwords is shown here:

```
agent.channels.ch-0.type = file
agent.channels.ch-0.capacity = 10000
agent.channels.ch-0.encryption.cipherProvider = AESCTRNOPADDING
agent.channels.ch-0.encryption.activeKey = key-1
agent.channels.ch-0.encryption.keyProvider = JCEKSFILE
agent.channels.ch-0.encryption.keyProvider.keyStoreFile = /path/to/my.keystore
agent.channels.ch-0.encryption.keyProvider.keyStorePasswordFile =
/path/to/my.keystore.password
agent.channels.ch-0.encryption.keyProvider.keys = key-0 key-1
agent.channels.ch-0.encryption.keyProvider.keys.key-0.passwordFile =
/path/to/key-0.password
agent.channels.ch-0.encryption.keyProvider.keys.key-1.passwordFile =
/path/to/key-1.password
```

## Troubleshooting

If the unlimited strength JCE policy files are not installed, an error similar to the following is printed in the flume.log:

```
07 Sep 2012 23:22:42,232 ERROR [lifecycleSupervisor-1-0]
(org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.getCipher:137) -
Unable to load key using transformation: AES/CTR/NoPadding; Warning: Maximum allowed
key length = 128 with the available JCE security policy files. Have you installed the
JCE unlimited strength jurisdiction policy files?
java.security.InvalidKeyException: Illegal key size
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.a(DashoA13*..)
at javax.crypto.Cipher.init(DashoA13*..)
at javax.crypto.Cipher.init(DashoA13*..)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.getCipher(AESCTRNoPaddingProvider.java:120)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider.access$200(AESCTRNoPaddingProvider.java:35)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$AESCTRNoPaddingDecryptor.<init>(AESCTRNoPaddingProvider.java:94)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$AESCTRNoPaddingDecryptor.<init>(AESCTRNoPaddingProvider.java:91)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$DecryptorBuilder.build(AESCTRNoPaddingProvider.java:66)
at
org.apache.flume.channel.file.encryption.AESCTRNoPaddingProvider$DecryptorBuilder.build(AESCTRNoPaddingProvider.java:62)
at
org.apache.flume.channel.file.encryption.CipherProviderFactory.getDecrypter(CipherProviderFactory.java:47)
at org.apache.flume.channel.file.LogFileV3$SequentialReader.<init>(LogFileV3.java:257)
at
org.apache.flume.channel.file.LogFileFactory.getSequentialReader(LogFileFactory.java:110)
at org.apache.flume.channel.file.ReplayHandler.replayLog(ReplayHandler.java:258)
at org.apache.flume.channel.file.Log.replay(Log.java:339)
at org.apache.flume.channel.file.FileChannel.start(FileChannel.java:260)
at
org.apache.flume.lifecycle.LifecycleSupervisor$MonitorRunnable.run(LifecycleSupervisor.java:236)
at java.util.concurrent.Executors$RunnableAdapter.call(Executors.java:441)
at java.util.concurrent.FutureTask$Sync.innerRunAndReset(FutureTask.java:317)
at java.util.concurrent.FutureTask.runAndReset(FutureTask.java:150)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.access$101(ScheduledThreadPoolExecutor.java:98)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.runPeriodic(ScheduledThreadPoolExecutor.java:180)
at
java.util.concurrent.ScheduledThreadPoolExecutor$ScheduledFutureTask.run(ScheduledThreadPoolExecutor.java:204)
at java.util.concurrent.ThreadPoolExecutor$Worker.runTask(ThreadPoolExecutor.java:886)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:908)
at java.lang.Thread.run(Thread.java:662)
```

### Viewing the Flume Documentation

For additional Flume documentation, see the [Flume User Guide](#) and the [Flume Developer Guide](#).

For additional information about Flume, see the [Apache Flume wiki](#).

# 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.

■ **Note:**

To see which version of Sqoop is shipping in CDH4, check the [CDH Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

See the following sections for information and instructions:

- [Upgrading from CDH3](#)
- [Upgrading from an Earlier CDH4 release](#)
- [Packaging](#)
- [Prerequisites](#)
- [Installing Packages](#)
- [Installing a Tarball](#)
- [Installing the JDBC Drivers](#)
- [Setting HADOOP\\_MAPRED\\_HOME for YARN](#)
- [Apache Sqoop Documentation](#)

## Upgrading Sqoop from CDH3 to CDH4

To upgrade Sqoop from CDH3 to CDH4, proceed as follows.

■ **Note:**

If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of Sqoop.

### Step 1: Remove the CDH3 version of Sqoop

To remove Sqoop on a Red Hat-compatible system:

```
$ sudo yum remove sqoop
```

To remove Sqoop on an Ubuntu or other Debian system:

```
$ sudo apt-get purge sqoop
```

■ **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

## Sqoop Installation

To remove Sqoop on a SLES system:

```
$ sudo zypper remove sqoop
```

### Step 2: Install the new version of Sqoop

Use one of the methods described below: [Installing the Sqoop RPM Packages](#) or [Installing the Sqoop Tarball](#).

■ **Important:**

During uninstall, the package manager renames any configuration files you have modified from <file> to <file>.rpm.save. During re-install, the package manager creates a new <file> with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

The upgrade is now complete.

## Upgrading Sqoop from an Earlier CDH4 release

These instructions assume that you are upgrading Sqoop as part of an upgrade to the latest CDH4 release, and have already performed the steps under [Upgrading from an Earlier CDH4 Release](#).

To upgrade Sqoop from an earlier CDH4 release, install the new version of Sqoop using one of the methods described below: [Installing the Sqoop RPM Packages](#) or [Installing the Sqoop Tarball](#).

■ **Important:**

During package upgrade, the package manager renames any configuration files you have modified from <file> to <file>.rpm.save, and creates a new <file> with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

## Sqoop Packaging

The packaging options for installing Sqoop are:

- RPM packages
- Tarball
- Debian packages

## Sqoop Prerequisites

- [An operating system supported by CDH4](#)
- [Oracle JDK](#)

## 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 packages consist of:

- `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` repository before using the following commands to install Sqoop. For instructions, see [CDH4 Installation](#).

**To install Sqoop on a Red Hat system:**

```
$ sudo yum install sqoop
```

**To install Sqoop on an Ubuntu or other Debian system:**

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

**To install Sqoop on a SLES system:**

```
$ sudo zypper install sqoop
```

If you have already configured CDH on your system, there is no further configuration necessary for Sqoop. 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 with YARN on a Unix-like system.

▪ **Important:**

Make sure you have read and understood [How Packaging Affects CDH4 Deployment](#) on page 22 before you proceed with a tarball installation.

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:**

## Sqoop Installation

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_MAPRED_HOME` are configured correctly. The variable `HADOOP_MAPRED_HOME` should point to the root directory of Hadoop installation. Optionally, if you intend to use any Hive or HBase related functionality, you must also make sure that they are installed and the variables `HIVE_HOME` and `HBASE_HOME` are configured correctly to point to the root directory of their respective installation.

## Installing the JDBC Drivers

Sqoop does not ship with third party JDBC drivers. You must download them separately and save them to the `/usr/lib/sqoop/lib/` directory. The following sections show how to install the most common JDBC Drivers.

### Installing the MySQL JDBC Driver

Download the MySQL JDBC driver from <http://www.mysql.com/downloads/connector/j/5.1.html>. You will need to sign up for an account if you don't already have one, and log in, before you can download it. Then copy it to the `/usr/lib/sqoop/lib/` directory. For example:

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar  
/usr/lib/sqoop/lib/
```

■ **Note:**

- At the time of publication, *version* was 5.1.25, but the version may have changed by the time you read this.
- If you installed Sqoop via Cloudera Manager, using parcels, copy the `.jar` file to `$HADOOP_CLASSPATH` instead of `/usr/lib/sqoop/lib/`.

### Installing the Oracle JDBC Driver

You can download the JDBC Driver from the Oracle website, for example <http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>. You must accept the license agreement before you can download the driver. Download the `ojdbc6.jar` file and copy it to `/usr/lib/sqoop/lib/` directory:

```
$ sudo cp ojdbc6.jar /usr/lib/sqoop/lib/
```

### Installing the Microsoft SQL Server JDBC Driver

Download the Microsoft SQL Server JDBC driver from <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774> and copy it to the `/usr/lib/sqoop/lib/` directory. For example:

```
$ curl -L  
'http://download.microsoft.com/download/0/2/A/02AAE597-3865-456C-AE7F-613F99F850A8/sqljdbc_4.0.2206.100_enu.tar.gz'  
| tar xz  
$ sudo cp sqljdbc_4.0/enu/sqljdbc4.jar /usr/lib/sqoop/lib/
```



## Installing the PostgreSQL JDBC Driver

Download the PostgreSQL JDBC driver from <http://jdbc.postgresql.org/download.html> and copy it to the `/usr/lib/sqoop/lib/` directory. For example:

```
$ curl -L 'http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar' -o  
postgresql-9.2-1002.jdbc4.jar  
$ sudo cp postgresql-9.2-1002.jdbc4.jar /usr/lib/sqoop/lib/
```

## Setting HADOOP\_MAPRED\_HOME for YARN

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

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

## Viewing the Sqoop Documentation

For additional documentation see the [Sqoop User Guide](#) and the [Sqoop Developer Guide](#).



# Sqoop 2 Installation

The following sections describe how to install and configure Sqoop 2:

- [About Sqoop 2](#)
- [Installing Sqoop 2](#)
- [Configuring Sqoop 2](#)
- [Starting, Stopping and Using the Server](#)
- [Apache Documentation](#)

## About Sqoop 2

Sqoop 2 is a server-based tool designed to transfer data between Hadoop and relational databases. You can use Sqoop 2 to import data from a relational database management system (RDBMS) such as MySQL or Oracle into the Hadoop Distributed File System (HDFS), transform the data with Hadoop MapReduce, and then export it back into an RDBMS.

- To find out more about Sqoop 2, see [Viewing the Sqoop 2 Documentation](#).
- To install Sqoop 2, follow the directions on [this page](#).

## Sqoop 2 Packaging

There are three packaging options for installing Sqoop 2:

- Tarball (.tgz) that contains both the Sqoop 2 server and the client.
- Separate RPM packages for Sqoop 2 server (`sqoop2-server`) and client (`sqoop2-client`)
- Separate Debian packages for Sqoop 2 server (`sqoop2-server`) and client (`sqoop2-client`)

## Installing Sqoop 2

Sqoop 2 is distributed as two separate packages: a client package (`sqoop2-client`) and a server package (`sqoop2-server`). Install the server package on one node in the cluster; because the Sqoop 2 server acts as a MapReduce client this node must have Hadoop installed and configured.

Install the client package on each node that will act as a client. A Sqoop 2 client will always connect to the Sqoop 2 server to perform any actions, so Hadoop does not need to be installed on the client nodes.

Depending on what you are planning to install, choose the appropriate package and install it using your preferred package manager application.

- **Note:** The Sqoop 2 packages can't be installed on the same machines as [Sqoop1](#) packages. However you can use both versions in the same Hadoop cluster by installing Sqoop1 and Sqoop 2 on different nodes.

To install the Sqoop 2 server package on a Red-Hat-compatible system:

- **Important:**  
If you have not already done so, install Cloudera's `yum`, `zypper`/YaST or `apt` repository before using the following commands to install Sqoop 2. For instructions, see [CDH4 Installation](#).

```
$ sudo yum install sqoop2-server
```

## Sqoop 2 Installation

To install the Sqoop 2 client package on an Red-Hat-compatible system:

```
$ sudo yum install sqoop2-client
```

To install the Sqoop 2 server package on a SLES system:

```
$ sudo zypper install sqoop2-server
```

To install the Sqoop 2 client package on an SLES system:

```
$ sudo zypper install sqoop2-client
```

To install the Sqoop 2 server package on an Ubuntu or Debian system:

```
$ sudo apt-get install sqoop2-server
```

To install the Sqoop 2 client package on an Ubuntu or Debian system:

```
$ sudo apt-get install sqoop2-client
```

- **Note:**

Installing the `sqoop2-server` package creates a `sqoop-server` service configured to start Sqoop 2 at system startup time.

You are now ready to configure Sqoop 2. See the [next section](#).

## Configuring Sqoop 2

This section explains how to configure the Sqoop 2 server.

### Configuring which Hadoop Version to Use

The Sqoop 2 client does not interact directly with Hadoop MapReduce, and so it does not require any MapReduce configuration.

The Sqoop 2 server can work with either MRv1 or YARN. **It cannot work with both simultaneously.**

The MapReduce version the Sqoop 2 server works with is determined by the `CATALINA_BASE` variable in the `/etc/defaults/sqoop2-server` file. By default, `CATALINA_BASE` is set to `/usr/lib/sqoop2/sqoop-server`. This setting configures the Sqoop 2 server to work with YARN. You need to change it to `/usr/lib/sqoop2/sqoop-server-0.20` to switch to MRv1.

Example `/etc/defaults/sqoop2-server` content to work with YARN:

```
CATALINA_BASE=/usr/lib/sqoop2/sqoop-server
```

Example `/etc/defaults/sqoop2-server` content to work with MRv1:

```
CATALINA_BASE=/usr/lib/sqoop2/sqoop-server-0.20
```

- **Note:**

`/etc/defaults/sqoop2-server` is loaded only once when the Sqoop 2 server starts. You must restart the Sqoop 2 server in order to apply any changes.

## Installing the JDBC Drivers

Sqoop 2 does not ship with third party JDBC drivers. You must download them separately and save them to the `/var/lib/sqoop2/` directory. The following sections show how to install the most common JDBC Drivers.

### Installing the MySQL JDBC Driver

Download the MySQL JDBC driver from <http://www.mysql.com/downloads/connector/j/5.1.html>. You will need to sign up for an account if you don't already have one, and log in, before you can download it. Then copy it to the `/var/lib/sqoop2/` directory. For example:

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar
/var/lib/sqoop2/
```

- **Note:** At the time of publication, *version* was 5.1.25, but the version may have changed by the time you read this.

### Installing the Oracle JDBC Driver

You can download the JDBC Driver from the Oracle website, for example <http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html>. You must accept the license agreement before you can download the driver. Download the `ojdbc6.jar` file and copy it to `/var/lib/sqoop2/` directory:

```
$ sudo cp ojdbc6.jar /var/lib/sqoop2/
```

### Installing the Microsoft SQL Server JDBC Driver

Download the Microsoft SQL Server JDBC driver from <http://www.microsoft.com/en-us/download/details.aspx?displaylang=en&id=11774> and copy it to the `/var/lib/sqoop2/` directory. For example:

```
$ curl -L 'http://download.microsoft.com/download/0/2/A/02AAE597-3865-456C-AE7F-613F99F850A8/sqljdbc_4.0.2206.100_enu.tar.gz'
| tar xz
$ sudo cp sqljdbc_4.0/enu/sqljdbc4.jar /var/lib/sqoop2/
```

### Installing the PostgreSQL JDBC Driver

Download the PostgreSQL JDBC driver from <http://jdbc.postgresql.org/download.html> and copy it to the `/var/lib/sqoop2/` directory. For example:

```
$ curl -L 'http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar' -o
postgresql-9.2-1002.jdbc4.jar
$ sudo cp postgresql-9.2-1002.jdbc4.jar /var/lib/sqoop2/
```

## Starting, Stopping, and Accessing the Sqoop 2 Server

### Starting the Sqoop 2 Server

After you have completed all of the required configuration steps, you can start Sqoop 2 server:

```
$ sudo /sbin/service sqoop2-server start
```

## Sqoop 2 Installation

### Stopping the Sqoop 2 Server

```
$ sudo /sbin/service sqoop2-server stop
```

### Checking that the Sqoop 2 Server has Started

You can verify whether the server has started correctly by connecting to its HTTP interface. The simplest way is to get the server version using following command:

```
$ wget -qO - localhost:12000/sqoop/version
```

You should get a text fragment in JSON format similar to the following:

```
{"version": "1.99.1-cdh4.4.0", ...}
```

### Accessing the Sqoop 2 Server with the Sqoop 2 Client

Start the Sqoop 2 client:

```
sqoop2
```

Identify the host where your server is running (we will use `localhost` in this example):

```
sqoop:000> set server --host localhost
```

Test the connection by running the command `show version --all` to obtain the version number from server. You should see output similar to the following:

```
sqoop:000> show version --all
server version:
  Sqoop 1.99.1-cdh4.4.0 revision ...
  Compiled by jenkins on ...
client version:
  Sqoop 1.99.1-cdh4.4.0 revision ...
  Compiled by jenkins on ...
Protocol version:
[1]
```

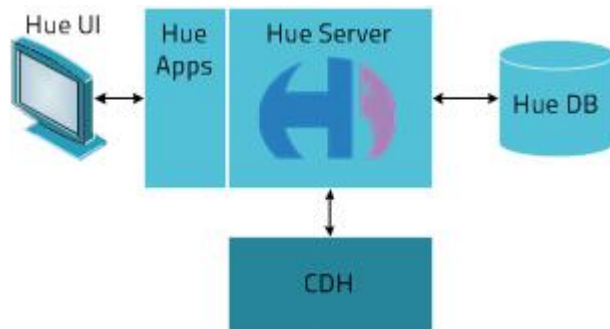
### Viewing the Sqoop 2 Documentation

For more information about Sqoop 2, see [https://blogs.apache.org/sqoop/entry/apache\\_sqoop\\_highlights\\_of\\_sqoop](https://blogs.apache.org/sqoop/entry/apache_sqoop_highlights_of_sqoop) and <http://archive.cloudera.com/cdh4/cdh/4/sqoop2>.

# Hue Installation

Hue is a suite of applications that provide web-based access to CDH components and a platform for building [custom applications](#).

The following figure illustrates how Hue works. Hue Server is a "container" web application that sits in between your CDH installation and the browser. It hosts the Hue applications and communicates with various servers that interface with CDH components.



The Hue Server uses a [database](#) to manage session, authentication, and Hue application data. For example, the Job Designer application stores job designs in the database.

Some Hue applications run Hue-specific daemon processes. For example, Beeswax runs a daemon (Beeswax Server) that keeps track of query states. Hue applications communicate with these daemons either by using Thrift or by exchanging state through the database.

In a CDH cluster, the Hue Server runs on a special node. For optimal performance, this should be one of the nodes within your cluster, though it can be a remote node as long as there are no overly restrictive firewalls. For small clusters of less than 10 nodes, you can use your existing master node as the Hue Server. In a pseudo-distributed installation, the Hue Server runs on the same machine as the rest of your CDH services.

## ■ Note: Install Cloudera Repository

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper`/`YaST` or `apt` repository, and install or upgrade CDH4 and make sure it is functioning correctly. For instructions, see [CDH4 Installation](#) and the instructions for [upgrading to CDH4](#) or [upgrading from an earlier CDH4 release](#).

## ■ Note: Running Services

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

Follow the instructions in the following sections to upgrade, install, configure, and administer Hue.

- [Supported Browsers](#)
- [Upgrading Hue](#)
- [Installing Hue](#)
- [Configuring CDH Components for Hue](#)

## Hue Installation

- [Configuring Your Firewall for Hue](#)
- [Configuring Hue](#)
- [Starting and Stopping the Server](#)
- [Administering Hue](#)
- [Hue User Guide](#)

## Supported Browsers

The Hue UI is supported on the following browsers:

- Windows: Chrome, Firefox 3.6+, Internet Explorer 9+, Safari 5+
- Linux: Chrome, Firefox 3.6+
- Mac: Chrome, Firefox 3.6+, Safari 5+

## Upgrading Hue

### ■ Note:

To see which version of Hue is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

### Upgrading Hue from CDH3 to CDH4

If you have already removed Hue as part of your upgrade to CDH4, skip to [Installing and Configuring Hue](#).

#### Step 1: Stop the Hue Server

See [Starting and Stopping the Hue Server](#).

#### Step 2: Uninstall the Old Version of Hue

- On RHEL systems:

```
$ sudo yum remove hue
```

- On SLES systems:

```
$ sudo zypper remove hue
```

- On Ubuntu or Debian systems:

```
sudo apt-get purge hue
```

### ■ Warning:

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

#### Step 3: Install Hue 2.x

Follow the instructions under [Installing Hue](#).



- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by `dpkg`](#).

#### Step 4: Start the Hue Server

See [Starting and Stopping the Hue Server](#).

## Upgrading Hue from an Earlier CDH4 Release to the Latest CDH4 Release

You can upgrade Hue either as part of an overall upgrade to the latest CDH4 release (see [Upgrading from an Earlier CDH4 Release](#)) or independently. To upgrade Hue from an earlier CDH4 release to the latest CDH4 release, proceed as follows.

#### Step 1: Stop the Hue Server

See [Starting and Stopping the Hue Server](#).

- **Warning:**

You **must** stop Hue. If Hue is running during the upgrade, the new version will not work correctly.

#### Step 2: Install the New Version of Hue

Follow the instructions under [Installing Hue](#).

- **Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by `dpkg`](#).

#### Step 3: Start the Hue Server

See [Starting and Stopping the Hue Server](#).

## Installing Hue

This section describes Hue installation and configuration on a cluster. The steps in this section apply whether you are installing on a single machine in pseudo-distributed mode, or on a cluster.

### Installing the Hue Packages

You must install the `hue-common` package on the machine where you will run the Hue Server. In addition, if you will be using Hue with MRv1, you must install the `hue-plugins` package on the system where you are running the JobTracker. (In pseudo-distributed mode, these will all be the same system.)

## Hue Installation

The `hue` meta-package installs the `hue-common` package and all the Hue applications; you also need to install `hue-server`, which contains the Hue start and stop scripts.

- **Note:** If you do not know which system your JobTracker is on, install the `hue-plugins` package on every node in the cluster.

### On RHEL systems:

- On the Hue Server machine, install the `hue` meta-package and the `hue-server` package:

```
$ sudo yum install hue hue-server
```

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the `hue-plugins` package:

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

### On SLES systems:

- On the Hue Server machine, install the `hue` meta-package and the `hue-server` package:

```
$ sudo zypper install hue hue-server
```

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the `hue-plugins` package:

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

### On Ubuntu or Debian systems:

- On the Hue Server machine, install the `hue` meta-package and the `hue-server` package:

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

- For MRv1: on the system that hosts the JobTracker, if different from the Hue server machine, install the `hue-plugins` package:

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

## Hue Dependencies

The following table shows the components that are dependencies for the different Hue applications and provides links to the installation guides for the required components that are not installed by default.

Component	Required	Applications
HDFS	Yes	Core, File Browser
MRv1	No	Job Browser, Job Designer*, Beeswax*, Oozie*
YARN	No	Job Browser, Job Designer*, Beeswax*, Oozie*
<a href="#">Oozie</a>	Yes	Job Designer, Oozie
<a href="#">Hive</a>	Yes	Beeswax, Metastore Manager
<a href="#">Cloudera Impala</a>	No	Cloudera Impala UI

Component	Required	Applications
<a href="#">HBase</a>	No	HBase Browser, Cloudera Impala UI, Shell
<a href="#">Pig</a>	No	Pig Editor, Shell, Oozie
<a href="#">Cloudera Search</a>	No	Solr Search
<a href="#">Sqoop</a>	No	Oozie
<a href="#">Sqoop 2</a>	No	Sqoop, Shell

[\*] Transitive dependency.

■ **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, see [hadoop.tmp.dir](#).

## Configuring CDH Components for Hue

To enable communication between the Hue Server and CDH components, you must make minor changes to your CDH installation by adding the properties described in this section to your CDH configuration files in `/etc/hadoop-0.20/conf/` or `/etc/hadoop/conf/`. If you are installing on a cluster, make these configuration changes on **each node** in your cluster.

### WebHDFS or HttpFS Configuration

Hue can use either of the following to access HDFS data:

- **WebHDFS** provides high-speed data transfer with good locality because clients talk directly to the DataNodes inside the Hadoop cluster.
- **HttpFS** is a proxy service appropriate for integration with external systems that are not behind the cluster's firewall.

Both WebHDFS and HttpFS use the HTTP REST API so they are fully interoperable, but Hue must be configured to use one or the other. For HDFS HA deployments, you must use HttpFS.

To configure Hue to use either WebHDFS or HttpFS, do the following steps:

**1. For WebHDFS only:**

- a. Add the following property in `hdfs-site.xml` to enable WebHDFS in the NameNode and DataNodes:

```
<property>
  <name>dfs.webhdfs.enabled</name>
  <value>true</value>
</property>
```

- b. Restart your HDFS cluster.

- 2.** Configure Hue as a proxy user for all other users and groups, meaning it may submit a request on behalf of any other user:

**WebHDFS:** Add to `core-site.xml`:

```
<!-- Hue WebHDFS proxy user setting -->
<property>
  <name>hadoop.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

**HttpFS:** Verify that `/etc/hadoop-httpfs/conf/httpfs-site.xml` has the following configuration:

```
<!-- Hue HttpFS proxy user setting -->
<property>
  <name>httpfs.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
<property>
  <name>httpfs.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

If the configuration is not present, add it to `/etc/hadoop-httpfs/conf/httpfs-site.xml` and restart the HttpFS daemon.

3. Verify that `core-site.xml` has the following configuration:

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

If the configuration is not present, add it to `/etc/hadoop/conf/core-site.xml` and restart Hadoop.

4. With root privileges, update `hadoop.hdfs_clusters.default.webhdfs_url` in `hue.ini` to point to the address of either WebHDFS or HttpFS.

```
[hadoop]
[[hdfs_clusters]]
[[[default]]]
# Use WebHdfs/HttpFs as the communication mechanism.
```

**WebHDFS:**

```
...
webhdfs_url=http://FQDN:50070/webhdfs/v1/
```

**HttpFS:**

```
...
webhdfs_url=http://FQDN:14000/webhdfs/v1/
```

- **Note:** If the `webhdfs_url` is uncommented and explicitly set to the empty value, Hue falls back to using the Thrift plugin used in Hue 1.x. This is not recommended.

## MRv1 Configuration

Hue communicates with the JobTracker via the Hue plugin, which is a `.jar` file that should be placed in your MapReduce `lib` directory.

- **Important:** The `hue-plugins` package installs the Hue plugins in your MapReduce `lib` directory, `/usr/lib/hadoop/lib`. If you are not using the package-based installation procedure, perform the following steps to install the Hue plugins.

If your JobTracker and Hue Server are located on the same host, copy the file over. If you are currently using CDH3, your MapReduce library directory might be in `/usr/lib/hadoop/lib`.

```
$ cd /usr/share/hue
$ cp desktop/libs/hadoop/java-lib/hue-plugins-*.jar /usr/lib/hadoop-0.20-mapreduce/lib
```

If your JobTracker runs on a different host, `scp` the Hue plugins `.jar` file to the JobTracker host.

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

```
<property>
  <name>jobtracker.thrift.address</name>
  <value>0.0.0.0:9290</value>
</property>
<property>
  <name>mapred.jobtracker.plugins</name>
  <value>org.apache.hadoop.thriftfs.ThriftJobTrackerPlugin</value>
  <description>Comma-separated list of jobtracker plug-ins to be activated.</description>
</property>
```

You can confirm that the plugins are running correctly by tailing the daemon logs:

```
$ tail --lines=500 /var/log/hadoop-0.20-mapreduce/hadoop*jobtracker*.log | grep
ThriftPlugin
2009-09-28 16:30:44,337 INFO org.apache.hadoop.thriftfs.ThriftPluginServer: Starting
Thrift server
2009-09-28 16:30:44,419 INFO org.apache.hadoop.thriftfs.ThriftPluginServer:
Thrift server listening on 0.0.0.0:9290
```

- **Note:** If you enable ACLs in the JobTracker, you must add users to the JobTracker `mapred.queue.default.acl-administer-jobs` property in order to allow Hue to display jobs in the Job Browser application. For example, to give the `hue` user access to the JobTracker, you would add the following property:

```
<property>
  <name>mapred.queue.default.acl-administer-jobs</name>
  <value>hue</value>
</property>
```

Repeat this for every user that requires access to the job details displayed by the JobTracker.

If you have any mapred queues besides "default", you must add a property for each queue:

```
<property>
  <name>mapred.queue.default.acl-administer-jobs</name>
  <value>hue</value>
</property>
<property>
  <name>mapred.queue.queue1.acl-administer-jobs</name>
  <value>hue</value>
</property>
<property>
  <name>mapred.queue.queue2.acl-administer-jobs</name>
  <value>hue</value>
</property>
```

## Hue Installation

### Oozie Configuration

In order to run DistCp, Streaming, Pig, Sqoop, and Hive jobs in Job Designer or the Oozie Editor/Dashboard application, you must make sure the Oozie shared libraries are installed for the correct version of MapReduce (MRv1 or YARN). See [Installing the Oozie ShareLib in Hadoop HDFS](#) for instructions.

### Hive Configuration

The Beeswax application helps you use Hive to query your data and depends on a Hive installation on your system. The Cloudera Impala application also depends on Hive.

- **Note:** When using Beeswax and Hive configured with the [embedded metastore](#) which is the default with Hue, the metastore DB should be owned by Hue (recommended) or writable to everybody:

```
sudo chown hue:hue -R /var/lib/hive/metastore/metastore_db
sudo chmod -R 777 /var/lib/hive/metastore/metastore_db
```

If not, Beeswax won't start and Hue Beeswax app will show 'Exception communicating with Hive Metastore Server at localhost:8003'

### Permissions

See [File System Permissions](#) in the Hive Installation section.

### No Existing Hive Installation

Familiarize yourself with the configuration options in `hive-site.xml`. See [Hive Installation](#). Having a `hive-site.xml` is optional but often useful, particularly on setting up a metastore. You can instruct Beeswax to locate it using the `hive_conf_dir` configuration variable.

### Existing Hive Installation

In the Hue configuration file `hue.ini`, modify `hive_conf_dir` to point to the directory containing `hive-site.xml`.

## Other Hadoop Settings

### HADOOP\_CLASSPATH

If you are setting `$HADOOP_CLASSPATH` in your `hadoop-env.sh`, be sure to set it in such a way that user-specified options are preserved. For example:

#### Correct:

```
# HADOOP_CLASSPATH=<your_additions>:$HADOOP_CLASSPATH
```

#### Incorrect:

```
# HADOOP_CLASSPATH=<your_additions>
```

This enables certain components of Hue to add to Hadoop's classpath using the environment variable.

### hadoop.tmp.dir

If your users are likely to be submitting jobs both using Hue and from the same machine via the command line interface, they will be doing so as the `hue` user when they are using Hue and via their own user account when they are using the command line. This leads to some contention on the directory specified by `hadoop.tmp.dir`, which defaults to `/tmp/hadoop- $\{user.name\}$` . Specifically, `hadoop.tmp.dir` is used to unpack JARs in

`/usr/lib/hadoop`. One work around to this is to set `hadoop.tmp.dir` to `/tmp/hadoop-{user.name}-{hue.suffix}` in the `core-site.xml` file:

```
<property>
  <name>hadoop.tmp.dir</name>
  <value>/tmp/hadoop-{user.name}-{hue.suffix}</value>
</property>
```

Unfortunately, when the `hue.suffix` variable is unset, you'll end up with directories named `/tmp/hadoop-user.name-{hue.suffix}` in `/tmp`. Despite that, Hue will still work.

■ **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.

## Configuring Your Firewall for Hue

Hue currently requires that the machines within your cluster can connect to each other freely over TCP. The machines outside your cluster must be able to open TCP port 8888 on the Hue Server (or the configured Hue web HTTP port) to interact with the system.


## Hue Configuration

This section describes configuration you perform in the Hue configuration file `hue.ini`. The location of the Hue configuration file varies depending on how Hue is installed. The location of the configuration file is displayed when you view the Hue configuration.


- **Note:** Only the root user can edit the Hue configuration file.

### Viewing the Hue Configuration

- **Note:** You must be a Hue superuser to view the Hue configuration.

When Hue detects an invalid configuration it displays the icon  on the top navigation bar. In addition, the Check Configuration page of the Quick Start wizard displays with information about misconfigured properties.

To view the Hue configuration, do one of the following:

- Click  and click the **Configuration** tab.
- Visit `http://myserver:port/dump_config`.

You can configure the Hue apps using the properties described in the sections:

- [Hue Server Configuration](#) on page 144
- [Beeswax Configuration](#) on page 145
- [Cloudera Impala Query UI Configuration](#) on page 145
- [Pig Editor Configuration](#) on page 146
- [Sqoop Configuration](#) on page 146
- [Job Browser Configuration](#) on page 146
- [Job Designer](#) on page 146

## Hue Installation

- [Oozie Editor/Dashboard Configuration](#) on page 146
- [Search Configuration](#) on page 147
- [Hue Shell Configuration](#) on page 147
- [HBase Configuration](#) on page 149
- [User Admin Configuration](#) on page 149
- [Configuring an LDAP Server for User Admin](#) on page 149
- [Hadoop Configuration](#) on page 152
- [Liboozie Configuration](#) on page 153

## Hue Server Configuration

This section describes Hue Server settings.

### Specifying the Hue Server HTTP Address

These configuration properties are under the `[desktop]` section in the Hue configuration file.

Hue includes two web servers, the CherryPy web server and the Spawning web server (configurable). You can use the following options to change the IP address and port that the web server listens on. The default setting is port 8888 on all configured IP addresses.

```
# Webserver listens on this address and port
http_host=0.0.0.0
http_port=8888
```

Hue defaults to using the Spawning web server, which is necessary for the Shell application. To revert to the CherryPy web server, use the following setting in the Hue configuration file:

```
use_cherrypy_server=true
```

Setting this to false causes Hue to use the Spawning web server.

### Specifying the Secret Key

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

1. Open the Hue configuration file.
2. In the `[desktop]` section, set the `secret_key` property to a long series of random characters (30 to 60 characters is recommended). For example,

```
secret_key=qpbdxoewsqlkhztybvfidtvwekftusgdlofbcfghaswuicmqp
```

- **Note:** If you don't specify a secret key, your session cookies will not be secure. Hue will run but it will also display error messages telling you to set the secret key.

## Authentication

By default, the first user who logs in to Hue can choose any username and password and automatically becomes an administrator. This user can create other user and administrator accounts. Hue users should correspond to the Linux users who will use Hue; make sure you use the same name as the Linux username.

By default, user information is stored in the Hue database. However, the authentication system is pluggable. You can configure authentication to use an LDAP directory (Active Directory or OpenLDAP) to perform the authentication, or you can import users and groups from an LDAP directory. See [Configuring an LDAP Server for User Admin](#) on page 149.

For more information, see [Hue SDK](#).



## Configuring the Hue Server for SSL

You can optionally configure Hue to serve over HTTPS. pyOpenSSL is now part of the Hue build and does not need to be installed manually. To configure SSL, perform the following steps from the root of your Hue installation path:

1. Configure Hue to use your private key by adding the following options to the Hue configuration file:

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

2. On a production system, you should have an appropriate key signed by a well-known Certificate Authority. If you're just testing, you can create a self-signed key using the `openssl` command that may be installed on your system:

```
# Create a key
$ openssl genrsa 1024 > host.key
# Create a self-signed certificate
$ openssl req -new -x509 -nodes -sha1 -key host.key > host.cert
```

- **Note:** Uploading files using the Hue File Browser over HTTPS requires using a proper SSL Certificate. Self-signed certificates don't work.

## Beeswax Configuration

In the `[beeswax]` section of the configuration file, you can optionally specify the following:

<code>beeswax_server_host</code>	The hostname or IP address of the Hive server. Default: localhost, and therefore only serves local IPC clients.
<code>beeswax_server_port</code>	The port of the Hive server.  If <code>server_interface</code> is set to <code>hiveserver2</code> , this should be set to the port that HiveServer2 is running on, which defaults to 10000. Default: 8002.
<code>hive_home_dir</code>	The base directory of the Hive installation.
<code>hive_conf_dir</code>	The directory containing the <code>hive-site.xml</code> Hive configuration file.
<code>beeswax_server_heapsize</code>	The heap size ( <code>-Xmx</code> ) of the Beeswax Server.
<code>server_interface</code>	The type of the Hive server that the application uses: <ul style="list-style-type: none"> <li>■ <code>beeswax</code></li> <li>■ <code>hiveserver2</code></li> </ul> Default: <code>beeswax</code> .

By default, Beeswax allows any user to see the saved queries of all other Beeswax users. You can restrict this by changing the setting the following property:

<code>share_saved_queries</code>	Set to <code>false</code> to restrict viewing of saved queries to the owner of the query or an administrator.
----------------------------------	---

## Cloudera Impala Query UI Configuration

In the `[impala]` section of the configuration file, you can optionally specify the following:

## Hue Installation

server_host	The hostname or IP address of the Impala Server. Default: localhost.
server_port	The port of the Impalad Server. Default: When using the beeswax interface, 21000. When using the HiveServer2 interface, 21050.
server_interface	The type of interface to use to communicate with Impalad Server: <ul style="list-style-type: none"><li>▪ beeswax</li><li>▪ hiveserver2</li></ul> Default: hiveserver2.

### Pig Editor Configuration

In the [pig] section of the configuration file, you can optionally specify the following:

remote_data_dir	Location on HDFS where the Pig examples are stored.
-----------------	---

### Sqoop Configuration

In the [sqoop] section of the configuration file, you can optionally specify the following:

server_url	The URL of the sqoop2 server.
------------	-------------------------------

### Job Browser Configuration

By default, any user can see submitted job information for all users. You can restrict viewing of submitted job information by optionally setting the following property under the [jobbrowser] section in the Hue configuration file:

share_jobs	Indicate that jobs should be shared with all users. If set to false, they will be visible only to the owner and administrators.
------------	---

### Job Designer

In the [jobsub] section of the configuration file, you can optionally specify the following:

remote_data_dir	Location in HDFS where the Job Designer examples and templates are stored.
-----------------	--

### Oozie Editor/Dashboard Configuration

By default, any user can see all workflows, coordinators, and bundles. You can restrict viewing of workflows, coordinators, and bundles by optionally specifying the following property under the [oozie] section of the Hue configuration file:

share_jobs	Indicate that workflows, coordinators, and bundles should be shared with all users. If set to false, they will be visible only to the owner and administrators.
oozie_jobs_count	Maximum number of Oozie workflows or coordinators or bundles to retrieve in one API call.
remote_data_dir	The location in HDFS where Oozie workflows are stored.

Also see [Liboozie Configuration](#) on page 153

## Search Configuration

In the `[search]` section of the configuration file, you can optionally specify the following:

<code>security_enabled</code>	Indicate whether Solr requires clients to perform Kerberos authentication.
<code>empty_query</code>	Query sent when no term is entered. Default: <code>* : *</code> .
<code>solr_url</code>	URL of the Solr server.

## Hue Shell Configuration

Hue includes the Shell application, which provides access to the Pig, HBase, and Sqoop 2 command-line shells. 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.

- **Note:** Flume 1.x does not provide a shell. However if you have Flume 0.9.x installed, you can access its shell through the Hue Shell application.
- **Note:** Pig Shell will not run unless you configure the `JAVA_HOME` environment variable in the Hue configuration file. For instructions, see [Hue Shell Configuration Properties](#) on page 147.

### Verifying Command-Line Shell Installations

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. For example:

**To verify that the Pig Shell (Grunt) is installed:**

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

**To verify that the HBase Shell is installed:**

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

**To verify that the Sqoop 2 Shell is installed:**

```
$ /usr/bin/sqoop2
```

### Hue Shell Configuration Properties

To add or remove shells, modify the Hue Shell configuration in the `[shell]` section of the Hue configuration file:

<code>shell_buffer_amount</code>	Optional. Amount of output to buffer for each shell in bytes. Defaults to 524288 (512 KiB) if not specified.
<code>shell_timeout</code>	Optional. Amount of time to keep shell subprocesses open when no open browsers refer to them. Defaults to 600 seconds if not specified.

<code>shell_write_buffer_limit</code>	Optional. Amount of pending commands to buffer for each shell, in bytes. Defaults to 10000 (10 KB) if not specified.
<code>shell_os_read_amount</code>	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.
<code>shell_delegation_token_dir</code>	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 <code>/tmp/hue_shell_delegation_tokens</code> if not specified.
<code>[[ shelltypes ]]</code>	This subsection groups the individual shell configurations.
<code>[[[ pig ]]]</code>	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 the beginning of a shell configuration section; each section can contain the following six parameters described in this table: <code>nice_name</code> , <code>command</code> , <code>help</code> , <code>environment</code> , and the value for the environment variable.
<code>nice_name = "Pig Shell (Grunt) "</code>	The user-facing name.
<code>command = "/usr/bin/pig -l /dev/null"</code>	The command to run to start the specified shell. The path to the binary must be an absolute path.
<code>help = "A platform for data exploration."</code>	Optional. A string that describes the shell.
<code>[[[[ environment ]]]]</code>	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.
<code>[[[[[ JAVA_HOME ]]]]]</code>	The name of the environment variable to set. For example, Pig requires <code>JAVA_HOME</code> to be set.
<code>value = /usr/lib/jvm/java-6-sun</code>	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.

### Unix User Accounts

To properly isolate subprocesses so as to guarantee security, each Hue user who is using the Shell subprocess must have a Unix user account. 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/build`) and execute one of the following commands in a terminal:

OS	Command
<b>sudo</b>	<pre>\$ sudo chown root:hue setuid \$ sudo chmod 4750 setuid \$ exit</pre>
<b>root</b>	<pre>\$ su # chown root:hue setuid # chmod 4750 setuid # exit</pre>

- **Important:** If you are running Hue Shell against a secure cluster, see [Running Hue Shell against a Secure Cluster](#) for security configuration information for Hue Shell.

### Hue Server Configuration

Older versions of Hue shipped with the CherryPy web server as the default Hue server. This is no longer the case starting with CDH3 Update 1. In order to configure the default Hue server, you must modify the Hue configuration file 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.

### HBase Configuration

In the `[hbase]` section of the configuration file, you can optionally specify the following:

<code>truncate_limit</code>	Hard limit of rows or columns per row fetched before truncating. Default: 500
<code>hbase_clusters</code>	Comma-separated list of HBase Thrift servers for clusters in the format of "(name  host:port)". Default: (Cluster  localhost:9090)

### User Admin Configuration

In the `[useradmin]` section of the configuration file, you can optionally specify the following:

<code>default_user_group</code>	The name of the group to which a manually created user is automatically assigned. Default: default.
---------------------------------	--

### Configuring an LDAP Server for User Admin

User Admin can interact with an LDAP server, such as Active Directory, in one of two ways:

- You can import user and group information from your current Active Directory infrastructure using the LDAP Import feature in the User Admin application. User authentication is then performed by User Admin based on the imported user and password information. You can then manage the imported users, along with any users you create directly in User Admin. See [Enabling Import of Users and Groups from an LDAP Directory](#) on page 150.
- You can configure User Admin to use an LDAP server as the authentication back end, which means users logging in to Hue will authenticate to the LDAP server, rather than against a username and password kept in User Admin. In this scenario, your users must all reside in the LDAP directory. See [Enabling the LDAP Server for User Authentication](#) on page 150 for further information.

### Enabling Import of Users and Groups from an LDAP Directory

User Admin can import users and groups from an Active Directory via the Lightweight Directory Authentication Protocol (LDAP). In order to use this feature, you must configure User Admin with a set of LDAP settings in the Hue configuration file.

- **Note:** If you import users from LDAP, you must set passwords for them manually; password information is not imported.

To enable LDAP import of users and groups:

1. In the Hue configuration file, configure the following properties in the `[[ldap]]` section:

Property	Description	Example
<code>base_dn</code>	The search base for finding users and groups.	<code>base_dn="DC=mycompany,DC=com"</code>
<code>nt_domain</code>	The NT domain to connect to (only for use with Active Directory).	<code>nt_domain=mycompany.com</code>
<code>ldap_url</code>	URL of the LDAP server.	<code>ldap_url=ldap://auth.mycompany.com</code>
<code>ldap_cert</code>	Path to certificate for authentication over TLS (optional).	<code>ldap_cert=/mycertsdir/myTLScert</code>
<code>bind_dn</code>	Distinguished name of the user to bind as – not necessary if the LDAP server supports anonymous searches.	<code>bind_dn="CN=ServiceAccount,DC=mycompany,DC=com"</code>
<code>bind_password</code>	Password of the bind user – not necessary if the LDAP server supports anonymous searches.	<code>bind_password=P@ssw0rd</code>

2. Configure the following properties in the `[[users]]` section:

Property	Description	Example
<code>user_filter</code>	Base filter for searching for users.	<code>user_filter="objectclass=*"</code>
<code>user_name_attr</code>	The username attribute in the LDAP schema.	<code>user_name_attr=sAMAccountName</code>

3. Configure the following properties in the `[[groups]]` section:

Property	Description	Example
<code>group_filter</code>	Base filter for searching for groups.	<code>group_filter="objectclass=*"</code>
<code>group_name_attr</code>	The username attribute in the LDAP schema.	<code>group_name_attr=cn</code>

- **Note:** If you provide a TLS certificate, it must be signed by a Certificate Authority that is trusted by the LDAP server.

### Enabling the LDAP Server for User Authentication

You can configure User Admin to use an LDAP server as the authentication back end, which means users logging in to Hue will authenticate to the LDAP server, rather than against usernames and passwords managed by User Admin.

- **Important:** Be aware that when you enable the LDAP back end for user authentication, user authentication by User Admin will be disabled. This means there will be no superuser accounts to log into Hue unless you take one of the following actions:
  - Import one or more superuser accounts from Active Directory and assign them superuser permission.
  - If you have already enabled the LDAP authentication back end, log into Hue using the LDAP back end, which will create a LDAP user. Then disable the LDAP authentication back end and use User Admin to give the superuser permission to the new LDAP user.

After assigning the superuser permission, enable the LDAP authentication back end.

#### To enable the LDAP server for user authentication:

1. In the Hue configuration file, configure the following properties in the `[[ldap]]` section:

Property	Description	Example
ldap_url	URL of the LDAP server, prefixed by <code>ldap://</code> or <code>ldaps://</code>	<code>ldap_url=ldap://auth.mycompany.com</code>
search_bind_authentication	Search bind authentication is disabled by default. To enable search bind authentication, the value of this property should be set to <code>true</code> . When using search bind, Hue will ignore the following <code>nt_domain</code> and <code>ldap_username_pattern</code> properties and discover Distinguished Names with LDAP search using the <code>user_name_attr</code> property.	<code>search_bind_authentication=true</code>
nt_domain	The NT domain over which the user connects (not strictly necessary if using <code>ldap_username_pattern</code> ).	<code>nt_domain=mycompany.com</code>
ldap_username_pattern	Pattern for searching for usernames – Use <code>&lt;username&gt;</code> for the username parameter. For use when using <code>LdapBackend</code> for Hue authentication	<code>ldap_username_pattern="uid=&lt;username&gt;,ou=People,dc=mycompany,dc=com"</code>

2. If you are using TLS or secure ports, add the following property to specify the path to a TLS certificate file:

Property	Description	Example
ldap_cert	Path to certificate for authentication over TLS. <div> <p>▪ <b>Note:</b> If you provide a TLS certificate, it must be signed by a Certificate Authority that is trusted by the LDAP server.</p> </div>	<code>ldap_cert=/mycertsdir/myTLScert</code>

3. In the `[[auth]]` sub-section inside `[desktop]` change the following:

backend	Change the setting of backend from <div> <code>backend=desktop.auth.backend.AllowFirstUserDjangoBackend</code> </div>
---------	---

	to
	backend=desktop.auth.backend.LdapBackend

## Hadoop Configuration

The following configuration variables are under the `[hadoop]` section in the Hue configuration file.

### HDFS Cluster Configuration

Hue currently supports only one HDFS cluster, which you define under the `[[hdfs_clusters]]` sub-section. The following properties are supported:

<code>[[[default]]]</code>	The section containing the default settings.
<code>fs_defaultfs</code>	The equivalent of <code>fs.defaultFS</code> (also referred to as <code>fs.default.name</code> ) in a Hadoop configuration.
<code>webhdfs_url</code>	The HttpFS URL. The default value is the HTTP port on the NameNode.
<code>hadoop_hdfs_home</code>	The home of your Hadoop HDFS installation. It is the root of the Hadoop untarred directory, or usually <code>/usr/lib/hadoop-hdfs</code> .
<code>hadoop_bin</code>	The HDFS Hadoop launcher script, which is usually <code>/usr/bin/hadoop</code> .
<code>hadoop_conf_dir</code>	The configuration directory of HDFS, typically <code>/etc/hadoop/conf</code> .

### MapReduce (MRv1) and YARN (MRv2) Cluster Configuration

Job Browser can display both MRv1 and MRv2 jobs, but must be configured to display one type at a time by specifying either `[[mapred_clusters]]` or `[[yarn_clusters]]` sections in the Hue configuration file.

The following MapReduce cluster properties are defined under the `[[mapred_clusters]]` sub-section:

<code>[[[default]]]</code>	The section containing the default settings.
<code>jobtracker_host</code>	The fully-qualified domain name of the host running the JobTracker.
<code>jobtracker_port</code>	The port for the JobTracker IPC service.
<code>submit_to</code>	If your Oozie is configured with to use a 0.20 MapReduce service, then set this to true. Indicate that Hue should submit jobs to this MapReduce cluster.
<code>hadoop_mapred_home</code>	The home directory of the Hadoop MapReduce installation. For CDH packages, the root of the Hadoop MRv1 untarred directory, the root of the Hadoop 2.0 untarred directory, <code>/usr/lib/hadoop-mapreduce</code> (for MRv1). If <code>submit_to</code> is true, the <code>\$HADOOP_MAPRED_HOME</code> for the Beeswax Server and child shell processes.
<code>hadoop_bin</code>	The MRv1 Hadoop launcher script, usually <code>/usr/bin/hadoop</code> .
<code>hadoop_conf_dir</code>	The configuration directory of the MRv1 service, typically <code>/etc/hadoop/conf</code> .

The following YARN cluster properties are defined under the under the `[[yarn_clusters]]` sub-section:



<code>[[[default]]]</code>	The section containing the default settings.
<code>resourcemanager_host</code>	The fully-qualified domain name of the host running the ResourceManager.
<code>resourcemanager_port</code>	The port for the ResourceManager IPC service.
<code>submit_to</code>	If your Oozie is configured to use a YARN cluster, then set this to true. Indicate that Hue should submit jobs to this YARN cluster.
<code>hadoop_mapred_home</code>	The home of the Hadoop MapReduce installation. For CDH packages, the root of the Hadoop 2.0 untarred directory, <code>/usr/lib/hadoop-mapreduce</code> (for YARN). If <code>submit_to</code> is true, the <code>\$HADOOP_MAPRED_HOME</code> for the Beeswax Server and child shell processes.
<code>hadoop_bin</code>	The YARN Hadoop launcher script, usually <code>/usr/bin/hadoop</code> .
<code>hadoop_conf_dir</code>	The configuration directory of the YARN service, typically <code>/etc/hadoop/conf</code> .

## Liboozie Configuration

In the `[liboozie]` section of the configuration file, you can optionally specify the following:

<code>security_enabled</code>	Indicate whether Oozie requires clients to perform Kerberos authentication.
<code>remote_deployment_dir</code>	The location in HDFS where the workflows and coordinators are deployed when submitted by a non-owner.
<code>oozie_url</code>	The URL of the Oozie server.

## Starting and Stopping the Hue Server

The `hue-server` package includes service scripts to start and stop the Hue Server.

**To start the Hue Server:**

```
$ sudo service hue start
```

**To stop the Hue Server:**

```
$ sudo service hue stop
```

## Administering Hue

The following sections contain details about managing and operating a Hue installation.

- [Anonymous Usage Data Collection](#) on page 154
- [Processes](#) on page 154
- [Logs](#) on page 154

## Hue Installation

### Anonymous Usage Data Collection

Hue tracks anonymised pages and application versions in order to gather information to help compare each application's usage levels. The data collected does not include any hostnames or IDs. For example, the data is of the form: `/2.3.0/pig, /2.5.0/beeswax/execute`.

For Hue 2.5.0 (and later), you can restrict this data collection by setting the `collect_usage` property to `false` under the `[desktop]` section in the Hue configuration file, `hue.ini`.

```
[desktop]
...
# Help improve Hue with anonymous usage analytics.
# Use Google Analytics to see how many times an application or specific section of an
  application is used, nothing more.
## collect_usage=false
```

If you are using an earlier version of Hue, disable data collection by navigating to Step 3 of Hue's Quick Start Wizard. Under **Anonymous usage analytics**, uncheck the **Check to enable usage analytics** checkbox.

### Processes

A script called `supervisor` manages all Hue processes. The supervisor is a watchdog process; its only purpose is to spawn and monitor other processes. A standard Hue installation starts and monitors the following processes:

- `runcpsserver` – a web server that provides the core web functionality of Hue
- `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`.

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.

### Logs

You can view the Hue logs in the `/var/log/hue` directory, where you can find:

- An `access.log` file, which contains a log for all requests against the Hue Server.
- A `supervisor.log` file, which contains log information for the supervisor process.
- A `supervisor.out` file, which contains the stdout and stderr for the supervisor process.
- A `.log` file for each supervised process described above, which contains the logs for that process.
- A `.out` file for each supervised process described above, which contains the stdout and stderr for that process.

If users on your cluster have problems running Hue, you can often find error messages in these log files.

#### Viewing Recent Log Messages

In addition to logging `INFO` level messages to the logs directory, the Hue Server keeps a small buffer of log messages at all levels in memory. The `DEBUG` level messages can sometimes be helpful in troubleshooting issues.

In the Hue UI you can view these messages by selecting the **Server Logs** tab in the **About** application. You can also view these logs by visiting `http://myserver:port/logs`.

### Hue Database

The Hue server requires an SQL database to store small amounts of data, including user account information as well as history of job submissions and Hive queries. The Hue server supports a lightweight embedded database and several types of external databases. If you elect to configure Hue to use an external database, upgrades may require more manual steps.

## Embedded Database

By default, Hue is configured to use the embedded database SQLite for this purpose, and should require no configuration or management by the administrator.

### Inspecting the Embedded Hue Database

The default SQLite database used by Hue is located in `/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 `sqlite3`, though `sqlite3` is useful for management or troubleshooting.

### Backing up the Embedded Hue Database

If you use the default embedded SQLite database, copy the `desktop.db` file to another node for backup. It is recommended that you back it up on a regular schedule, and also that you back it up before any upgrade to a new version of Hue.

## External Database

Although SQLite is the default database, some advanced users may prefer to have Hue access an external database. Hue supports MySQL, PostgreSQL, and Oracle. See [Supported Databases](#) for the supported versions.

- **Note:** In the instructions that follow, dumping the database and editing the JSON objects is only necessary if you have data in SQLite that you need to migrate. If you don't need to migrate data from SQLite, you can skip those steps.

### Configuring the Hue Server to Store Data in MySQL

1. Shut down the Hue server if it is running.
2. Dump the existing database data to a text file. Note that using the `.json` extension is required.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field.
4. Start the Hue server.
5. Install the MySQL client developer package.

OS	Command
RHEL	<code>\$ sudo yum install mysql-devel</code>
SLES	<code>\$ sudo zypper install mysql-devel</code>
Ubuntu or Debian	<code>\$ sudo apt-get install libmysqlclient-dev</code>

## 6. Install the MySQL connector.

OS	Command
RHEL	\$ sudo yum install mysql-connector-java
SLES	\$ sudo zypper install mysql-connector-java
Ubuntu or Debian	\$ sudo apt-get install libmysql-java

## 7. Install and start MySQL.

OS	Command
RHEL	\$ sudo yum install mysql-server
SLES	\$ sudo zypper install mysql \$ sudo zypper install libmysqlclient_r15
Ubuntu or Debian	\$ sudo apt-get install mysql-server

## 8. Change the /etc/my.cnf file as follows:

```
[mysqld]
datadir=/var/lib/mysql
socket=/var/lib/mysql/mysql.sock
bind-address=<ip-address>
default-storage-engine=MyISAM
```

## 9. Start the mysql daemon.

OS	Command
RHEL	\$ sudo service mysqld start
SLES and Ubuntu or Debian	\$ sudo service mysql start

## 10. Configure MySQL to use a strong password. In the following procedure, your current root password is blank. Press the **Enter** key when you're prompted for the root password.

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

## 11. Configure MySQL to start at boot.

OS	Command
RHEL	\$ sudo /sbin/chkconfig mysqld on \$ sudo /sbin/chkconfig --list mysqld

OS	Command
	mysqld 0:off 1:off 2:on 3:on 4:on 5:on 6:off
SLES	\$ sudo chkconfig --add mysql
Ubuntu or Debian	\$ sudo chkconfig mysql on

12. Create the Hue database and grant privileges to a hue user to manage the database.

```
mysql> create database hue;
Query OK, 1 row affected (0.01 sec)
mysql> grant all on hue.* to 'hue'@'localhost' identified by '<secretpassword>';
Query OK, 0 rows affected (0.00 sec)
```

13. Open the Hue configuration file in a text editor.

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

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

15. As the hue user, load the existing data and create the necessary database tables.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
$ mysql -uhue -p<secretpassword>
mysql > SHOW CREATE TABLE auth_permission;
```

16. (InnoDB only) Drop the foreign key.

```
mysql > ALTER TABLE auth_permission DROP FOREIGN KEY content_type_id_refs_id_XXXXXX;
```

17. Delete the rows in the `django_content_type` table.

```
mysql > DELETE FROM hue.django_content_type;
```

18. Load the data.

```
$ <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

19. (InnoDB only) Add the foreign key.

```
$ mysql -uhue -p<secretpassword>
mysql > ALTER TABLE auth_permission ADD FOREIGN KEY (`content_type_id`) REFERENCES
`django_content_type` (`id`);
```

## Configuring the Hue Server to Store Data in PostgreSQL

1. Shut down the Hue server if it is running.
2. Dump the existing database data to a text file. Note that using the `.json` extension is required.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json
```

3. Open `<some-temporary-file>.json` and remove all JSON objects with `useradmin.userprofile` in the `model` field.

#### 4. Install required packages.

OS	Command
RHEL	\$ sudo yum install postgresql-devel gcc python-devel
SLES	\$ sudo zypper install postgresql-devel gcc python-devel
Ubuntu or Debian	\$ sudo apt-get install postgresql-devel gcc python-devel

#### 5. Install the module that provides the connector to PostgreSQL.

```
sudo -u hue <HUE_HOME>/build/env/bin/pip install setuptools
sudo -u hue <HUE_HOME>/build/env/bin/pip install psycopg2
```

#### 6. Install the PostgreSQL server.

OS	Command
RHEL	\$ sudo yum install postgresql-server
SLES	\$ sudo zypper install postgresql-server
Ubuntu or Debian	\$ sudo apt-get install postgresql

#### 7. Initialize the data directories:

```
$ service postgresql initdb
```

#### 8. Configure client authentication.

- Edit `/var/lib/pgsql/data/pg_hba.conf`.
- Set the authentication methods for local to `trust` and for host to `password` and add the following line at the end.

```
host hue hue 0.0.0.0/0 md5
```

#### 9. Start the PostgreSQL server.

```
$ su - postgres
# /usr/bin/postgres -D /var/lib/pgsql/data > logfile 2>&1 &
```

#### 10. Configure PostgreSQL to listen on all network interfaces.

Edit `/var/lib/pgsql/data/postgresql.conf` and set `listen_addresses`:

```
listen_addresses = '0.0.0.0'      # Listen on all addresses
```

#### 11. Create the hue database and grant privileges to a hue user to manage the database.

```
# psql -U postgres
postgres=# create database hue;
postgres=# \c hue;
You are now connected to database 'hue'.
postgres=# create user hue with password '<secretpassword>';
postgres=# grant all privileges on database hue to hue;
postgres=# \q
```

12. Restart the PostgreSQL server.

```
$ sudo service postgresql restart
```

13. Verify connectivity.

```
psql -h localhost -U hue -d hue
Password for user hue: <secretpassword>
```

14. Configure the PostgreSQL server to start at boot.

OS	Command
RHEL	<pre>\$ sudo /sbin/chkconfig postgresql on \$ sudo /sbin/chkconfig --list postgresql postgresql          0:off  1:off  2:on   3:on   4:on   5:on                   6:off</pre>
SLES	<pre>\$ sudo chkconfig --add postgresql</pre>
Ubuntu or Debian	<pre>\$ sudo chkconfig postgresql on</pre>

15. Open the Hue configuration file in a text editor.

16. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your PostgreSQL setup).

```
host=localhost
port=5432
engine=postgresql_psycpg2
user=hue
password=<secretpassword>
name=hue
```

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

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
```

18. Determine the foreign key ID.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# \d auth_permission;
```

19. Drop the foreign key that you retrieved in the previous step.

```
postgres=# ALTER TABLE auth_permission DROP CONSTRAINT
content_type_id_refs_id_<XXXXXX>;
```

20. Delete the rows in the `django_content_type` table.

```
postgres=# TRUNCATE django_content_type CASCADE;
```

21. Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

22. Add back the foreign key you dropped.

```
bash# su - postgres
$ psql -h localhost -U hue -d hue
postgres=# ALTER TABLE auth_permission ADD CONSTRAINT
```

```
content_type_id_refs_id_<XXXXXX> FOREIGN KEY (content_type_id) REFERENCES
django_content_type(id) DEFERRABLE INITIALLY DEFERRED;
```

### Configuring the Hue Server to Store Data in Oracle

1. Ensure Python 2.6 or newer is installed on the server Hue is running on.
2. Download the Oracle client libraries at [Instant Client for Linux x86-64](#) Version 11.1.0.7.0, Basic and SDK (with headers) zip files to the same directory.
3. Unzip the zip files.
4. Set environment variables to reference the libraries.

```
$ export ORACLE_HOME=<download directory>
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ORACLE_HOME
```

5. Create a symbolic link for the shared object:

```
$ cd $ORACLE_HOME
$ ln -sf libclntsh.so.11.1 libclntsh.so
```

6. Install the Python Oracle library.

```
$ <HUE_HOME>/build/env/bin/pip install cx_Oracle
```

7. Upgrade django south.

```
$ <HUE_HOME>/build/env/bin/pip install south --upgrade
```

8. Get a data dump by executing:

```
$ <HUE_HOME>/build/env/bin/hue dumpdata > <some-temporary-file>.json --indent 2
```

9. Open the Hue configuration file in a text editor.
10. Directly below the `[[database]]` section under the `[desktop]` line, add the following options (and modify accordingly for your Oracle setup):

```
host=localhost
port=1521
engine=oracle
user=hue
password=<secretpassword>
name=<SID of the Oracle database, for example, 'XE'>
```

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

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue syncdb --noinput
```

12. Migrate the Hue database.

```
$ <HUE_HOME>/build/env/bin/hue migrate
```

13. Generate statements to delete all data from Oracle tables:

```
> SELECT 'DELETE FROM ' || '.' || table_name || ';' FROM user_tables;
```

14. Run the statements generated in the preceding step.



15. Load the data.

```
$ sudo -u hue <HUE_HOME>/build/env/bin/hue loaddata <some-temporary-file>.json
```

## Viewing the Hue User Guide

For additional information about Hue, see the [Hue User Guide](#).



## Pig Installation

Apache Pig enables you to analyze large amounts of data using Pig's query language called Pig Latin. Pig Latin queries run in a distributed way on a Hadoop cluster.

■ **Important:**

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands to install or upgrade Pig. For instructions, see [CDH4 Installation](#).

Use the following sections to install or upgrade Pig:

- [Upgrading Pig](#)
- [Installing Pig](#)
- [Using Pig with HBase](#)
- [Installing DataFu](#)
- [Apache Pig Documentation](#)

## Upgrading Pig

■ **Note:**

To see which version of Pig is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [Release Notes](#).

### Upgrading Pig from CDH3 to CDH4

To upgrade Pig to CDH4:

■ **Note:**

If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of Pig.

#### Step 1: Remove Pig

1. Exit the Grunt shell and make sure no Pig scripts are running.
2. Remove the CDH3 version of Pig

**To remove Pig On Red Hat-compatible systems:**

```
$ sudo yum remove hadoop-pig
```

**To remove Pig on SLES systems:**

```
$ sudo zypper remove hadoop-pig
```

**To remove Pig on Ubuntu and other Debian systems:**

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

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

### Step 2: Install the new version

Follow the instructions in the next section, [Installing Pig](#).

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

### Incompatible Changes as of the Pig 0.7.0 Release

Pig 0.7.0 contained several changes that are not backward-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) so that they 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](#).

## Upgrading Pig from an Earlier CDH4 release

The instructions that follow assume that you are upgrading Pig as part of a CDH4 upgrade, and have already performed the steps under [Upgrading from an Earlier CDH4 Release](#).

To upgrade Pig from an earlier CDH4 release:

1. Exit the Grunt shell and make sure no Pig scripts are running.
2. Install the new version, following the instructions in the next section, [Installing Pig](#).

- **Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

## Installing Pig

To install Pig On Red Hat-compatible systems:

```
$ sudo yum install pig
```

**To install Pig on SLES systems:**

```
$ sudo zypper install pig
```

**To install Pig on Ubuntu and other Debian systems:**

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

- **Note:**

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

**To start the Grunt Shell (MRv1):**

Use commands similar to the following, replacing the <component\_version> strings with the current HBase, ZooKeeper and CDH versions.

```
$ export PIG_CONF_DIR=/usr/lib/pig/conf
$ export
PIG_CLASSPATH=/usr/lib/hbase/hbase-0.9.2-cdh4.4.0-security.jar:/usr/lib/zookeeper/zookeeper-3.4.5-cdh4.4.0.jar
$ pig
2012-02-08 23:39:41,819 [main] INFO org.apache.pig.Main - Logging error messages to:
/home/arvind/pig-0.9.2-cdh4b1/bin/pig_1328773181817.log
2012-02-08 23:39:41,994 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop
file system at: hdfs://localhost/
...
grunt>
```

Here's an example.

```
$ export PIG_CONF_DIR=/usr/lib/pig/conf
$ export
PIG_CLASSPATH=/usr/lib/hbase/hbase-0.94.6-cdh4.4.0-security.jar:/usr/lib/zookeeper/zookeeper-3.4.5-cdh4.4.0.jar
$ pig
2012-02-08 23:39:41,819 [main] INFO org.apache.pig.Main - Logging error messages to:
/home/arvind/pig-0.9.2-cdh4b1/bin/pig_1328773181817.log
2012-02-08 23:39:41,994 [main] INFO
org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop
file system at: hdfs://localhost/
...
grunt>
```

**To start the Grunt Shell (YARN):**

- **Important:**

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

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

Use the following commands, replacing the <component\_version> string with the current HBase, ZooKeeper and CDH version numbers.

```
$ export PIG_CONF_DIR=/usr/lib/pig/conf
$ export
PIG_CLASSPATH=/usr/lib/hbase/hbase-0.9.2-cdh4.4.0-security.jar:/usr/lib/zookeeper/zookeeper-3.4.5-cdh4.4.0.jar
$ pig
...
grunt>
```

## Pig Installation

For example,

```
$ export PIG_CONF_DIR=/usr/lib/pig/conf
$ export
PIG_CLASSPATH=/usr/lib/hbase/hbase-0.94.6-cdh4.4.0-security.jar:/usr/lib/zookeeper/zookeeper-3.4.5-cdh4.4.0.jar
$ pig
...
grunt>
```

To verify that the input and output directories from the example `grep` job exist (see [Installing CDH4 on a Single Linux Node in Pseudo-distributed Mode](#)), list an HDFS directory from the Grunt Shell:

```
grunt> ls
hdfs://localhost/user/joe/input <dir>
hdfs://localhost/user/joe/output <dir>
```

To run a `grep` example job using Pig for `grep` inputs:

```
grunt> A = LOAD 'input';
grunt> B = FILTER A BY $0 MATCHES '.*dfs[a-z.]+.*';
grunt> DUMP B;
```

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

## Using Pig with HBase

To allow Pig scripts to use HBase, add the following statement to the top of each script. Replace the `<component_version>` strings with the current HBase, ZooKeeper and CDH version numbers.

```
register /usr/lib/zookeeper/zookeeper-<ZooKeeper_version>-cdh<CDH_version>.jar
register /usr/lib/hbase/hbase-<HBase_version>-cdh<CDH_version>-security.jar
```

For example,

```
register /usr/lib/zookeeper/zookeeper-3.4.5-cdh4.4.0.jar
register /usr/lib/hbase/hbase-0.94.6-cdh4.4.0-security.jar
```

## Installing DataFu

DataFu is a collection of Apache Pig UDFs (User-Defined Functions) for statistical evaluation that were developed by LinkedIn and have now been open sourced under an Apache 2.0 license.

**To use DataFu:**

1. Install the DataFu package:

Operating system	Install command
Red-Hat-compatible	<code>sudo yum install pig-udf</code>
SLES	<code>sudo zypper install pig-udf-datafu</code>
Debian or Ubuntu	<code>sudo apt-get install pig-udf-datafu</code>

This puts the `datafu-0.0.4-cdh4.4.0.jar` file in `/usr/lib/pig`.

2. Register the JAR. Replace the `<component_version>` string with the current DataFu and CDH version numbers.

```
REGISTER /usr/lib/pig/datafu-<DataFu_version>-cdh<CDH_version>.jar
```

For example,

```
REGISTER /usr/lib/pig/datafu-0.0.4-cdh4.4.0.jar
```

A number of usage examples and other information are available at <https://github.com/linkedin/datafu>.

## Viewing the Pig Documentation

For additional Pig documentation, see <http://archive.cloudera.com/cdh4/cdh/4/pig>.





# Oozie Installation

- [About Oozie](#)
- [Packaging](#)
- [Prerequisites](#)
- [Upgrading Oozie](#)
- [Installing Oozie](#)
- [Configuring Oozie](#)
- [Starting, Stopping, and Using the Server](#)
- [Configuring Failover](#)
- [Apache Oozie Documentation](#)

## About Oozie

Apache Oozie Workflow Scheduler for Hadoop is a workflow and coordination service for managing Apache Hadoop jobs:

- Oozie Workflow jobs are Directed Acyclical Graphs (DAGs) of *actions*; *actions* are typically Hadoop jobs (MapReduce, Streaming, Pipes, Pig, Hive, Sqoop, etc).
- Oozie Coordinator jobs trigger recurrent Workflow jobs based on time (frequency) and data availability.
- Oozie Bundle jobs are sets of Coordinator jobs managed as a single job.

Oozie is an extensible, scalable and data-aware service that you can use to orchestrate dependencies among jobs running on Hadoop.

- To find out more about Oozie, see <http://archive.cloudera.com/cdh4/cdh/4/oozie/>.
- To install or upgrade Oozie, follow the directions on this page.

### ■ Note: Running Services

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

## Oozie Packaging

There are two packaging options for installing Oozie:

- Separate RPM packages for the Oozie server (`oozie`) and client (`oozie-client`)
- Separate Debian packages for the Oozie server (`oozie`) and client (`oozie-client`)

You can also [download an Oozie tarball](#).

## Oozie Prerequisites

- Prerequisites for installing Oozie server:

## Oozie Installation

- [An operating system supported by CDH4](#)
  - [Oracle JDK](#)
  - [A supported database](#) if you are not planning to use the default (Derby).
- Prerequisites for installing Oozie client:
    - [Oracle JDK](#)

■ **Note:**

- To see which version of Oozie is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).
- The following CDH4 versions of Hadoop work with the CDH4 version of Oozie: Beta 2, CDH4.0.0.

## Upgrading Oozie

Follow these instructions to upgrade Oozie to CDH4 from RPM or Debian Packages.

■ **Note: Before you start:**

Make sure there are no workflows in `RUNNING` or `SUSPENDED` status; otherwise the [database upgrade](#) will fail and you will have to reinstall Oozie CDH3 to complete or kill those running workflows.

## Upgrading Oozie from CDH3 to CDH4

To upgrade Oozie from CDH3 to CDH4, back up the configuration files and database, uninstall the CDH3 version and then install and configure the CDH4 version. Proceed as follows.

■ **Note:**

If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of Oozie.

■ **Note:**

**Important: Ubuntu and Debian upgrades**

When you uninstall CDH3 Oozie on Ubuntu and Debian systems, the contents of `/var/lib/oozie` are removed, leaving a bare directory. This can cause the Oozie upgrade to CDH4 to fail. To prevent this, either copy the database files to another location and restore them after the uninstall, or recreate them after the uninstall. Make sure you do this before starting the re-install.

### Step 1: Remove Oozie

1. Back up the Oozie configuration files in `/etc/oozie` and the Oozie database. For convenience you may want to save Oozie configuration files in your home directory; you will need them after installing the new version of Oozie.
2. Stop the Oozie Server.

**To stop the Oozie Server:**

```
sudo service oozie stop
```

### 3. Uninstall Oozie.

To uninstall Oozie, run the appropriate command on each host:

- On Red Hat-compatible systems:

```
$ sudo yum remove oozie-client
```

- On SLES systems:

```
$ sudo zypper remove oozie-client
```

- On Ubuntu or Debian systems:

```
sudo apt-get purge oozie-client
```

- Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

### Step 2: Install Oozie

Follow the procedure under [Installing Oozie](#) and then proceed to [Configuring Oozie after Upgrading from CDH3](#). For packaging information, see [Oozie Packaging](#).

- Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

## Upgrading Oozie from an Earlier CDH4 Release to the Latest CDH4 Release

The steps that follow assume you are upgrading Oozie as part of an overall upgrade to the latest CDH4 release and have already performed the steps under [Upgrading from an Earlier CDH4 Release](#).

To upgrade Oozie to the latest CDH4 release, proceed as follows.

### Step 1: Back Up the Configuration

Back up the Oozie configuration files in `/etc/oozie` and the Oozie database.

For convenience you may want to save Oozie configuration files in your home directory; you will need them after installing the new version of Oozie.

### Step 2: Stop the Oozie Server.

To stop the Oozie Server:

```
sudo service oozie stop
```

### Step 3: Install Oozie

Follow the procedure under [Installing Oozie](#) and then proceed to [Configuring Oozie after Upgrading from an Earlier CDH4 Release](#).

- **Important:**

During package upgrade, the package manager renames any configuration files you have modified from <file> to <file>.rpm.save, and creates a new <file> with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

## Installing Oozie

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

- **Note:**

The Oozie server package, `oozie`, is preconfigured to work with MRv1. To configure the Oozie server to work with Hadoop MapReduce YARN, see [Configuring the Hadoop Version to Use](#).

- **Important:**

If you have not already done so, install Cloudera's Yum, zypper/YaST or Apt repository before using the following commands to install Oozie. For instructions, see [CDH4 Installation](#).

**To install the Oozie server package on an Ubuntu and other Debian system:**

```
$ sudo apt-get install oozie
```

**To install the Oozie client package on an Ubuntu and other Debian system:**

```
$ sudo apt-get install oozie-client
```

**To install the Oozie server package on a Red Hat-compatible system:**

```
$ sudo yum install oozie
```

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

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

**To install the Oozie server package on a SLES system:**

```
$ sudo zypper install oozie
```

**To install the Oozie client package on a SLES system:**

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

■ **Note:**

Installing the `oozie` package creates an `oozie` service configured to start Oozie at system startup time.

You are now ready to configure Oozie. See the [next section](#).

## Configuring Oozie

This section explains how to [configure which Hadoop version to use](#), and provides separate procedures for each of the following:

- [Configuring Oozie after Upgrading from CDH3](#) on page 173
- [Configuring Oozie after Upgrading from an Earlier CDH4 Release](#) on page 176
- [Configuring Oozie after a New Installation](#) on page 177

### Configuring which Hadoop Version to Use

The Oozie client does not interact directly with Hadoop MapReduce, and so it does not require any MapReduce configuration.

The Oozie server can work with either MRv1 or YARN. **It cannot work with both simultaneously.**

The MapReduce version Oozie server works with is determined by the `CATALINA_BASE` variable in the `/etc/oozie/conf/oozie-env.sh` file. By default, `CATALINA_BASE` is set to `/usr/lib/oozie/oozie-server-0.20`. This setting configures the Oozie server to work with MRv1.

To configure the Oozie server to work with YARN instead, set `CATALINA_BASE` to `/usr/lib/oozie/oozie-server`.

### Configuring Oozie after Upgrading from CDH3

■ **Note:**

If you are installing Oozie for the first time, skip this section and proceed with [Configuring Oozie after a New Installation](#).

#### Step 1: Update Configuration Files

1. Edit the the new Oozie CDH4 `oozie-site.xml`, and set all customizable properties to the values you set in the CDH3 `oozie-site.xml`:

■ **Important:**

- DO NOT copy over the CDH3 configuration files into the CDH4 configuration directory.
- The configuration property names for the database settings have changed between Oozie CDH3 and Oozie CDH4: the prefix for these names has changed from `oozie.service.StoreService.*` to `oozie.service.JPAService.*`. Make sure you use the new prefix.

2. If necessary do the same for the `oozie-log4j.properties`, `oozie-env.sh` and the `adminusers.txt` files.

### Step 2: Upgrade the Database

- **Important:**

- Do not proceed before you have edited the configuration files as instructed in [Step 1](#).
- Before running the database upgrade tool, copy or symlink the MySQL JDBC driver JAR into the `/var/lib/oozie/` directory.

Oozie CDH4 provides a command-line tool to perform the database schema and data upgrade that is required when you upgrade Oozie from CDH3 to CDH4. The tool uses Oozie configuration files to connect to the database and perform the upgrade.

The database upgrade tool works in two modes: it can do the upgrade in the database or it can produce an SQL script that a database administrator can run manually. If you use the tool to perform the upgrade, you must do it as a database user who has permissions to run DDL operations in the Oozie database.

#### To run the Oozie database upgrade tool against the database:

- **Important:**

This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh upgrade -run
```

You will see output such as this:

```
Validate DB Connection.
DONE
Check DB schema exists
DONE
Check OOZIE_SYS table does not exist
DONE
Verify there are not active Workflow Jobs
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE
Upgrade COORD_JOBS new columns default values.
DONE
Upgrade COORD_JOBS & COORD_ACTIONS status values.
DONE
Table 'WF_ACTIONS' column 'execution_path', length changed to 1024
DONE

Oozie DB has been upgraded to Oozie version '3.3.2-cdh4.4.0'

The SQL commands have been written to: /tmp/ooziedb-5737263881793872034.sql
```

#### To create the upgrade script:

- **Important:**

This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh -sqlfile <SCRIPT>
```

For example:

```
$ bin/ooziedb.sh upgrade -sqlfile oozie-upgrade.sql
```

You should see output such as the following:

```
Validate DB Connection.
DONE
Check DB schema exists
DONE
Check OOZIE_SYS table does not exist
DONE
Verify there are not active Workflow Jobs
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE
Upgrade COORD_JOBS new columns default values.
DONE
Upgrade COORD_JOBS & COORD_ACTIONS status values.
DONE
Table 'WF_ACTIONS' column 'execution_path', length changed to 1024
DONE

Oozie DB has been upgraded to Oozie version '3.3.2-cdh4.4.0'

The SQL commands have been written to: oozie-upgrade.sql

WARN: The SQL commands have NOT been executed, you must use the '-run' option
```

■ **Important:**

If you used the `-sqlfile` option instead of `-run`, Oozie database schema has not been upgraded. You need to run the `oozie-upgrade` script against your database.

### Step 3: Upgrade the Oozie Sharelib

■ **Important:**

This step is required; CDH4 Oozie does not work with CDH3 shared libraries.

CDH4 Oozie has a new shared library which bundles CDH4 JAR files for streaming, DistCp and for Pig, Hive and Sqoop.

The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:

- The shared library file for MRv1 is `oozie-sharelib.tar.gz`.
- The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.

1. Delete the Oozie shared libraries from HDFS. For example:

```
$ sudo -u oozie hadoop fs -rmr /user/oozie/share
```

■ **Note:**

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

## Oozie Installation

2. Expand the Oozie CDH4 shared libraries in a local temp directory and copy them to HDFS. For example:

```
$ 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
```

- **Important:**

If you are installing Oozie to work with YARN use `oozie-sharelib-yarn.tar.gz` instead.

- **Note:**

If the current shared libraries are in another location, make sure you use this other location when you run the above commands, and if necessary edit the `oozie-site.xml` configuration file to point to the right location.

### Step 4: Start the Oozie Server

Now you can start Oozie:

```
$ sudo service oozie start
```

Check Oozie's `oozie.log` to verify that Oozie has started successfully.

### Step 5: Upgrade the Oozie Client

Although older Oozie clients work with the new Oozie server, you need to install the new version of the Oozie client in order to use all the functionality of the Oozie server.

To upgrade the Oozie client, if you have not already done so, follow the steps under [Installing Oozie](#).

## Configuring Oozie after Upgrading from an Earlier CDH4 Release

- **Note:**

If you are installing Oozie for the first time, skip this section and proceed with [Configuring Oozie after a New Installation](#).

### Step 1: Update Configuration Files

1. Edit the the new Oozie CDH4 `oozie-site.xml`, and set all customizable properties to the values you set in the previous `oozie-site.xml`.
2. If necessary do the same for the `oozie-log4j.properties`, `oozie-env.sh` and the `adminusers.txt` files.

### Step 2: Upgrade the Oozie Sharelib

- **Important:**

This step is required; the current version of Oozie does not work with shared libraries from an earlier version.

The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:



- The shared library file for MRv1 is `oozie-sharelib.tar.gz`.
- The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.

1. Delete the Oozie shared libraries from HDFS. For example:

```
$ sudo -u oozie hadoop fs -rmr /user/oozie/share
```

■ **Note:**

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

2. Expand the Oozie CDH4 shared libraries in a local temp directory and copy them to HDFS. For example:

```
$ 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
```

■ **Important:**

If you are installing Oozie to work with YARN use `oozie-sharelib-yarn.tar.gz` instead.

■ **Note:**

If the current shared libraries are in another location, make sure you use this other location when you run the above commands, and if necessary edit the `oozie-site.xml` configuration file to point to the right location.

### Step 3: Start the Oozie Server

Now you can start Oozie:

```
$ sudo service oozie start
```

Check Oozie's `oozie.log` to verify that Oozie has started successfully.

### Step 4: Upgrade the Oozie Client

Although older Oozie clients work with the new Oozie server, you need to install the new version of the Oozie client in order to use all the functionality of the Oozie server.

To upgrade the Oozie client, if you have not already done so, follow the steps under [Installing Oozie](#).

## Configuring Oozie after a New Installation

■ **Note:**

Follow the instructions in this section if you are installing Oozie for the first time. If you are upgrading Oozie from CDH3 or from an earlier CDH4 release, skip this section and choose the appropriate instructions under [Configuring Oozie](#).

## Oozie Installation

When you install Oozie from an RPM or Debian package, Oozie server creates all configuration, documentation, and runtime files in the standard Linux directories, as follows.

Type of File	Where Installed
binaries	<code>/usr/lib/oozie/</code>
configuration	<code>/etc/oozie/conf/</code>
documentation	for SLES: <code>/usr/share/doc/packages/oozie/</code> for other platforms: <code>/usr/share/doc/oozie/</code>
examples TAR.GZ	for SLES: <code>/usr/share/doc/packages/oozie/</code> for other platforms: <code>/usr/share/doc/oozie/</code>
sharelib TAR.GZ	<code>/usr/lib/oozie/</code>
data	<code>/var/lib/oozie/</code>
logs	<code>/var/log/oozie/</code>
temp	<code>/var/tmp/oozie/</code>
PID file	<code>/var/run/oozie/</code>

### Deciding which Database to Use

Oozie has a built-in Derby database, but Cloudera recommends that you use a [Postgres](#), [MySQL](#), or [Oracle](#) database instead, for the following reasons:

- Derby runs in embedded mode and it is not possible to monitor its health.
- It is not clear how to implement a live backup strategy for the embedded Derby database, though it may be possible.
- Under load, Cloudera has observed locks and rollbacks with the embedded Derby database which don't happen with server-based databases.

### Configuring Oozie to Use Postgres

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

#### Step 1: Install PostgreSQL 8.4.x or 9.0.x.

■ **Note:**

See [CDH4 Requirements and Supported Versions](#) for tested versions.

#### 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.UTF8'
LC_CTYPE = 'en_US.UTF8'
CONNECTION LIMIT = -1;
CREATE DATABASE

postgres=# \q

```

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

1. Edit the `postgresql.conf` file and set the `listen_addresses` property to `*`, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the `standard_conforming_strings` property is set to `off`.
2. Edit the 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/PostgreSQL/8.4/data
```

### Step 5: Configure Oozie to use Postgres.

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

```

...
<property>
  <name>oozie.service.JPAService.jdbc.driver</name>
  <value>org.postgresql.Driver</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.url</name>
  <value>jdbc:postgresql://localhost:5432/oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.username</name>
  <value>oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.password</name>
  <value>oozie</value>
</property>
...

```

#### ■ Note:

In the JDBC URL property, replace `localhost` with the hostname where 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.

## Configuring Oozie to Use MySQL

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

### Step 1: Install and start MySQL 5.x

- **Note:**

See [CDH4 Requirements and Supported Versions](#) for tested versions.

### Step 2: Create the Oozie database and Oozie MySQL user.

For example, using the MySQL `mysql` command-line tool:

```
$ mysql -u root -p
Enter password: *****

mysql> create database oozie;
Query OK, 1 row affected (0.03 sec)

mysql> grant all privileges on oozie.* to 'oozie'@'localhost' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> grant all privileges on oozie.* to 'oozie'@'%' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> exit
Bye
```

### Step 3: Configure Oozie to use MySQL.

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

```
...
<property>
  <name>oozie.service.JPAService.jdbc.driver</name>
  <value>com.mysql.jdbc.Driver</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.url</name>
  <value>jdbc:mysql://localhost:3306/oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.username</name>
  <value>oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.password</name>
  <value>oozie</value>
</property>
...
```

- **Note:**

In the JDBC URL property, replace `localhost` with the hostname where MySQL is running.

### Step 4: Add the MySQL JDBC driver JAR to Oozie.

Copy or symlink the MySQL JDBC driver JAR into the `/var/lib/oozie/` directory.

- **Note:**

You must manually download the MySQL JDBC driver JAR file.

### Configuring Oozie to use Oracle

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

- Note:**

See [CDH4 Requirements and Supported Versions](#) for tested versions.

### Step 1: Install and start Oracle 11g.

### Step 2: Create the Oozie Oracle user.

For example, using the Oracle `sqlplus` command-line tool:

```
$ sqlplus system@localhost

Enter password: *****

SQL> create user oozie identified by oozie default tablespace users temporary tablespace
temp;

User created.

SQL> grant all privileges to oozie;

Grant succeeded.

SQL> exit

$
```

### Step 3: Configure Oozie to use Oracle.

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

```
...
<property>
  <name>oozie.service.JPAService.jdbc.driver</name>
  <value>oracle.jdbc.driver.OracleDriver</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.url</name>
  <value>jdbc:oracle:thin:@localhost:1521:oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.username</name>
  <value>oozie</value>
</property>
<property>
  <name>oozie.service.JPAService.jdbc.password</name>
  <value>oozie</value>
</property>
...
```

- Note:**

In the JDBC URL property, replace `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.

Copy or symlink the Oracle JDBC driver JAR into the `/var/lib/oozie/` directory.

- Note:**

You must manually download the Oracle JDBC driver JAR file.

### Creating the Oozie Database Schema

After configuring Oozie database information and creating the corresponding database, create the Oozie database schema. Oozie provides a database tool for this purpose.

- **Note:**

The Oozie database tool uses Oozie configuration files to connect to the database to perform the schema creation; before you use the tool, make you have created a database and configured Oozie to work with it as described above.

The Oozie database tool works in 2 modes: it can create the database, or it can produce an SQL script that a database administrator can run to create the database manually. If you use the tool to create the database schema, you must have the permissions needed to execute DDL operations.

**To run the Oozie database tool against the database:**

- **Important:**

This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh create -run
```

You should see output such as the following:

```
Validate DB Connection.
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
DONE
DONE
Create OOZIE_SYS table
DONE

Oozie DB has been created for Oozie version '3.3.2-cdh4.4.0'

The SQL commands have been written to: /tmp/ooziedb-5737263881793872034.sql
```

**To create the upgrade script:**

- **Important:**

This step must be done as the `oozie` Unix user, otherwise Oozie may fail to start or work properly because of incorrect file permissions.

Run `/usr/lib/oozie/bin/ooziedb.sh create -sqlfile <SCRIPT>`. For example:

```
$ sudo -u oozie /usr/lib/oozie/bin/ooziedb.sh create -sqlfile oozie-create.sql
```

You should see output such as the following:

```
Validate DB Connection.
DONE
Check DB schema does not exist
DONE
Check OOZIE_SYS table does not exist
DONE
Create SQL schema
```

```

DONE
DONE
Create OOZIE_SYS table
DONE

Oozie DB has been created for Oozie version '3.3.2-cdh4.4.0'

The SQL commands have been written to: oozie-create.sql

WARN: The SQL commands have NOT been executed, you must use the '-run' option

```

■ **Important:**

If you used the `-sqlfile` option instead of `-run`, Oozie database schema has not been created. You need to run the `oozie-create.sql` script against your database.

### Enabling the Oozie Web Console

To enable Oozie's web console, you must download and add the ExtJS library to the Oozie server. *If you have not already done this*, proceed as follows.

#### Step 1: Download the Library

Download the ExtJS version 2.2 library from <http://archive.cloudera.com/gplextras/misc/ext-2.2.zip> and place it a convenient location.

#### Step 2: Install the Library

Extract the `ext-2.2.zip` file into `/var/lib/oozie`.

### Configuring Oozie with Kerberos Security

To configure Oozie with Kerberos security, see [Oozie Security Configuration](#).

### Installing the Oozie ShareLib in Hadoop HDFS

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

The Oozie installation bundles two shared libraries, one for MRv1 and one for YARN. Make sure you install the right one for the MapReduce version you are using:

- The shared library file for MRv1 is `oozie-sharelib.tar.gz`.
- The shared library file for YARN is `oozie-sharelib-yarn.tar.gz`.

■ **Important:**

If Hadoop is configured with Kerberos security enabled, you must first configure Oozie with Kerberos Authentication. For instructions, see [Oozie Security Configuration](#). Before running the commands in the following instructions, you must run the `sudo -u oozie kinit -k -t /etc/oozie/oozie.keytab` and `kinit -k hdfs` commands. Then, instead of using commands in the form `sudo -u <user> <command>`, use just `<command>`; for example, `$ hadoop fs -mkdir /user/oozie`

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

```

$ sudo -u hdfs hadoop fs -mkdir /user/oozie
$ sudo -u hdfs hadoop fs -chown oozie:oozie /user/oozie
$ mkdir /tmp/ooziesharelib
$ cd /tmp/ooziesharelib

```

## Oozie Installation

```
$ tar xzf /usr/lib/oozie/oozie-sharelib.tar.gz
$ sudo -u oozie hadoop fs -put share /user/oozie/share
```

- **Important:**

If you are installing Oozie to work with YARN use `oozie-sharelib-yarn.tar.gz` instead.

### Configuring Support for Oozie Uber JARs

An **uber JAR** is a JAR that contains other JARs with dependencies in a `lib/` folder inside the JAR. Beginning with CDH4.1, you can configure the cluster to handle uber JARs properly for the MapReduce action (as long as it does not include any streaming or pipes) by setting the following property in the `oozie-site.xml` file:

```
...
<property>
  <name>oozie.action.mapreduce.uber.jar.enable</name>
  <value>true</value>
...
```

When this property is set, users can use the `oozie.mapreduce.uber.jar` configuration property in their MapReduce workflows to notify Oozie that the specified JAR file is an uber JAR.

### Configuring Oozie to Run against a Federated Cluster

To run Oozie against a federated HDFS cluster using ViewFS, configure the `oozie.service.HadoopAccessorService.supported.filesystems` property in `oozie-site.xml` as follows:

```
<property>
  <name>oozie.service.HadoopAccessorService.supported.filesystems</name>
  <value>hdfs,viewfs</value>
</property>
```

## Starting, Stopping, and Accessing the Oozie Server

### Starting the Oozie Server

After you have completed *all* of the required configuration steps, you can start Oozie:

```
$ sudo service oozie start
```

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

- **Note:**

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

### Stopping the Oozie Server

```
$ sudo service oozie stop
```



## Accessing the Oozie Server with the Oozie Client

The Oozie client is a command-line utility that interacts with the Oozie server via the Oozie web-services API.

Use the `/usr/bin/oozie` script to run the Oozie client.

For example, if you want to invoke the client on the same machine where the Oozie server is running:

```
$ oozie admin -oozie http://localhost:11000/oozie -status
System mode: NORMAL
```

To make it convenient to use this utility, set the environment variable `OOZIE_URL` to point to the URL of the Oozie server. Then you can skip the `-oozie` option.

For example, if you want to invoke the client on the same machine where the Oozie server is running, set the `OOZIE_URL` to `http://localhost:11000/oozie`.

```
$ export OOZIE_URL=http://localhost:11000/oozie
$ oozie admin -version
Oozie server build version: 3.1.3-cdh4.0.0
```

### ■ Important:

If Oozie is configured with Kerberos Security enabled:

- You must have a Kerberos session running. For example, you can start a session by running the `kinit` command.
- **Do not** use `localhost` as in the above examples.

As with every service that uses Kerberos, Oozie has a Kerberos *principal* in the form `<SERVICE>/<HOSTNAME>@<REALM>`. In a Kerberos configuration, you **must** use the `<HOSTNAME>` value in the Kerberos principal to specify the Oozie server; for example, if the `<HOSTNAME>` in the principal is `myoozieserver.mydomain.com`, set `OOZIE_URL` as follows:

```
$ export OOZIE_URL=http://myoozieserver.mydomain.com:11000/oozie
```

If you use an alternate hostname or the IP address of the service, Oozie will not work properly.

## Accessing the Oozie Server with a Browser

If you have enabled the Oozie web console by adding the ExtJS library, you can connect to the the console at `http://<OOZIE_HOSTNAME>:11000/oozie`.

### ■ Note:

If the Oozie server is configured to use Kerberos HTTP SPNEGO Authentication, you must use a web browser that supports Kerberos HTTP SPNEGO (for example, Firefox or Internet Explorer).

## Configuring Oozie Failover ( hot/cold )

1. Set up your database for High Availability (see the database documentation for details).

### ■ Note:

Oozie database configuration properties may need special configuration (see the JDBC driver documentation for details).

## Oozie Installation

2. Configure Oozie on two or more servers:
3. These servers should be configured identically
4. Set the `OOZIE_HTTP_HOSTNAME` variable in `oozie-env.sh` to the Load Balancer or Virtual IP address (see step 3)
5. Only one of the Oozie servers should be started (the *hot* server).
6. Use either a Virtual IP Address or Load Balancer to direct traffic to the hot server.
7. Access Oozie via the Virtual IP or Load Balancer address.

### Points to note

- The Virtual IP Address or Load Balancer can be used to periodically check the health of the hot server.
- If something is wrong, you can shut down the hot server, start the cold server, and redirect the Virtual IP Address or Load Balancer to the new hot server.
- This can all be automated with a script, but a false positive indicating the hot server is down will cause problems, so test your script carefully.
- There will be no data loss.
- Any running workflows will continue from where they left off.
- It takes only about 15 seconds to start the Oozie server.

See also [Configuring Oozie to use HDFS HA](#).

## Viewing the Oozie Documentation

For additional Oozie documentation, see <http://archive.cloudera.com/cdh4/cdh/4/oozie/>.

# Hive Installation

## ■ **Note: Install Cloudera Repository**

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper`/`YaST` or `apt` repository, and install or upgrade CDH4 and make sure it is functioning correctly. For instructions, see [CDH4 Installation](#) and the instructions for [upgrading to CDH4](#) or [upgrading from an earlier CDH4 release](#).

## ■ **Note: Running Services**

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

Use the following sections to install, update, and configure Hive:

- [About Hive](#)
- [Upgrading Hive](#)
- [Installing Hive](#)
- [Configuring the Metastore](#)
- [Configuring HiveServer2](#)
- [Starting the Metastore](#)
- [File System Permissions](#)
- [Starting, Stopping, and Using HiveServer2](#)
- [Starting the Hive Console](#)
- [Using Hive with HBase](#)
- [Installing the JDBC on Clients](#)
- [Setting HADOOP\\_MAPRED\\_HOME for YARN](#)
- [Configuring the Metastore for HDFS HA](#)
- [Troubleshooting](#)
- [Apache Hive Documentation](#)

## About Hive

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

Install Hive on your client machine(s) from which you submit jobs; you do not need to install it on the nodes in your Hadoop cluster.

## HiveServer2

As of CDH4.1, you can deploy [HiveServer2](#), an improved version of HiveServer that supports a new Thrift API tailored for JDBC and ODBC clients, Kerberos authentication, and multi-client concurrency. There is also a new CLI for HiveServer2 named [Beeline](#).

Cloudera recommends you install HiveServer2, and use it whenever possible. (You can still use the original HiveServer when you need to, and run it concurrently with HiveServer2; see [Configuring HiveServer2](#)).

## Upgrading Hive

Upgrade Hive on all the hosts on which it is running: servers and clients.

- **Note:** To see which version of Hive is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

### Upgrading Hive from CDH3 to CDH4

- **Note:**  
If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of Hive.

#### Step 1: Remove Hive

- **Warning:**  
You **must** make sure no Hive processes are running. If Hive processes are running during the upgrade, the new version will not work correctly.

1. Exit the Hive console and make sure no Hive scripts are running.
2. Stop any HiveServer processes that are running. If HiveServer is running as a daemon, use the following command to stop it:

```
$ sudo service hive-server stop
```

If HiveServer is running from the command line, stop it with <CTRL>-c.

3. Stop the metastore. If the metastore is running as a daemon, use the following command to stop it:

```
$ sudo service hive-metastore stop
```

If the metastore is running from the command line, stop it with <CTRL>-c.

4. Remove Hive:

- **Note:**  
The following examples show how to uninstall Hive packages on a CDH3 system. Note that CDH3 and CDH4 use different names for the Hive packages: in CDH3 Hive, package names begin with the prefix `hadoop-hive`, while in CDH4 they begin with the prefix `hive`. (If you are already running CDH4 and upgrading to the latest version, you do not need to remove Hive: see [Upgrading Hive from an Earlier Version of CDH4](#).)

```
$ sudo yum remove hadoop-hive
```

To remove Hive on SLES systems:

```
$ sudo zypper remove hadoop-hive
```

**To remove Hive on Ubuntu and Debian systems:**

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

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, **DO NOT PROCEED** before backing them up.

**Step 2: Install the new Hive version on all hosts (Hive servers and clients)**

See [Installing Hive](#).

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

**Step 3: Configure the Hive Metastore**

You must configure the Hive metastore and initialize the service before you start the Hive Console. See [Configuring the Hive Metastore](#) for detailed instructions.

**Step 4: Upgrade the Metastore Schema**

The current version of CDH4 includes changes in the Hive metastore schema. If you have been using Hive 0.9 or earlier, you must upgrade the Hive metastore schema after you install the new version of Hive but before you start Hive. To do this, run the appropriate schema upgrade scripts in

```
/usr/lib/hive/scripts/metastore/upgrade/:
```

- Schema upgrade scripts from 0.7 to 0.8 and from 0.8 to 0.9 for Derby, MySQL, and PostgreSQL
- 0.8 and 0.9 schema scripts for Oracle, but no upgrade scripts (you will need to create your own)
- Schema upgrade scripts from 0.9 to 0.10 for Derby, MySQL, PostgreSQL and Oracle

- **Note:** To upgrade Hive from CDH3 to CDH4, you must upgrade the schema to 0.8, then to 0.9, and then to 0.10.

- **Important:**

- Cloudera strongly encourages you to make a backup copy of your metastore database before running the upgrade scripts. You will need this backup copy if you run into problems during the upgrade or need to downgrade to a previous version.
- You **must** upgrade the metastore schema before starting Hive after the upgrade. Failure to do so may result in metastore corruption.
- To run a script, you must first `cd` to the directory that script is in: that is `/usr/lib/hive/scripts/metastore/upgrade/<database>`.

## Hive Installation

For more information about upgrading the schema, see the README in `/usr/lib/hive/scripts/metastore/upgrade/`.

### Step 5: Configure HiveServer2

HiveServer2 is an improved version of the original HiveServer (HiveServer1). Cloudera recommends using HiveServer2 instead of HiveServer1 as long as you do not depend directly on HiveServer1's Thrift API. Some configuration is required before you initialize HiveServer2; see [Configuring HiveServer2](#) for details.

■ **Note:**

**If you need to run HiveServer1**

You can continue to run HiveServer1 on CDH4.1 and later if you need it for backward compatibility; for example, you may have existing Perl and Python scripts that use the native HiveServer1 Thrift bindings. You can install and run HiveServer1 and HiveServer2 concurrently on the same system; see [Running HiveServer2 and HiveServer Concurrently](#).

### Step 6: Upgrade Scripts, etc., for HiveServer2 (if necessary)

If you have been running HiveServer1, you may need to make some minor modifications to your client-side scripts and applications when you upgrade:

- HiveServer1 does not support concurrent connections, so many customers run a dedicated instance of HiveServer1 for each client. These can now be replaced by a single instance of HiveServer2.
- HiveServer2 uses a different connection URL and driver class for the JDBC driver. If you have existing scripts that use JDBC to communicate with HiveServer1, you can modify these scripts to work with HiveServer2 by changing the JDBC driver URL from `jdbc:hive://hostname:port` to `jdbc:hive2://hostname:port`, and by changing the JDBC driver class name from `org.apache.hive.jdbc.HiveDriver` to `org.apache.hive.jdbc.HiveDriver`.

### Step 7: Start the Metastore, HiveServer2, and Beeline

See:

- [Starting the Metastore](#)
- [Starting HiveServer2](#)
- [Using Beeline](#)

## Upgrading Hive from an Earlier Version of CDH4

The instructions that follow assume that you are upgrading Hive as part of an upgrade to CDH4, and have already performed the steps under [Upgrading to CDH4](#).

■ **Important:**

If you are currently running Hive under MRv1, check for the following property and value in `/etc/mapred/conf/mapred-site.xml`:

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
</property>
```

Remove this property before you proceed; otherwise Hive queries spawned from MapReduce jobs will fail with a null pointer exception (NPE).

To upgrade Hive from an earlier version of CDH4, proceed as follows.

## Step 1: Stop all Hive Processes and Daemons

- **Warning:**

You **must** make sure no Hive processes are running. If Hive processes are running during the upgrade, the new version will not work correctly.

1. Exit the Hive console and make sure no Hive scripts are running.
2. Stop any HiveServer processes that are running. If HiveServer is running as a daemon, use the following command to stop it:

```
$ sudo service hive-server stop
```

If HiveServer is running from the command line, stop it with <CTRL>-c.

3. Stop any HiveServer2 processes that are running. If HiveServer2 is running as a daemon, use the following command to stop it:

```
$ sudo service hive-server2 stop
```

If HiveServer2 is running from the command line, stop it with <CTRL>-c.

4. Stop the metastore. If the metastore is running as a daemon, use the following command to stop it:

```
$ sudo service hive-metastore stop
```

If the metastore is running from the command line, stop it with <CTRL>-c.

## Step 2: Install the new Hive version on all hosts (Hive servers and clients)

See [Installing Hive](#).

## Step 3: Verify that the Hive Metastore is Properly Configured

See [Configuring the Hive Metastore](#) for detailed instructions.

## Step 4: Upgrade the Metastore Schema

The current version of CDH4 includes changes in the Hive metastore schema. If you have been using Hive 0.9 or earlier, you must upgrade the Hive metastore schema after you install the new version of Hive but before you start Hive. To do this, run the appropriate schema upgrade scripts in

`/usr/lib/hive/scripts/metastore/upgrade/:`

- Schema upgrade scripts from 0.7 to 0.8 and from 0.8 to 0.9 for Derby, MySQL, and PostgreSQL
- 0.8 and 0.9 schema scripts for Oracle, but no upgrade scripts (you will need to create your own)
- Schema upgrade scripts from 0.9 to 0.10 for Derby, MySQL, PostgreSQL and Oracle

- **Important:**

- Cloudera strongly encourages you to make a backup copy of your metastore database before running the upgrade scripts. You will need this backup copy if you run into problems during the upgrade or need to downgrade to a previous version.
- You **must** upgrade the metastore schema before starting Hive. Failure to do so may result in metastore corruption.
- To run a script, you must first `cd` to the directory that script is in: that is `/usr/lib/hive/scripts/metastore/upgrade/<database>`.

## Hive Installation

For more information about upgrading the schema, see the README in `/usr/lib/hive/scripts/metastore/upgrade/`.

### Step 5: Configure HiveServer2

HiveServer2 is an improved version of the original HiveServer (HiveServer1). Cloudera recommends using HiveServer2 instead of HiveServer1 in most cases. Some configuration is required before you initialize HiveServer2; see [Configuring HiveServer2](#) for details.

■ **Note:**

**If you need to run HiveServer1**

You can continue to run HiveServer1 on CDH4.1 and later if you need it for backward compatibility; for example, you may have existing Perl and Python scripts that use the native HiveServer1 Thrift bindings. You can install and run HiveServer1 and HiveServer2 concurrently on the same systems; see [Running HiveServer2 and HiveServer Concurrently](#).

### Step 6: Upgrade Scripts, etc., for HiveServer2 (if necessary)

If you have been running HiveServer1, you may need to make some minor modifications to your client-side scripts and applications when you upgrade:

- HiveServer1 does not support concurrent connections, so many customers run a dedicated instance of HiveServer1 for each client. These can now be replaced by a single instance of HiveServer 2.
- HiveServer2 uses a different connection URL and driver class for the JDBC driver; scripts may need to be modified to use the new version.
- Perl and Python scripts that use the native HiveServer1 Thrift bindings may need to be modified to use the HiveServer2 Thrift bindings.

### Step 7: Start the Metastore, HiveServer2, and Beeline

See:

- [Starting the Metastore](#)
- [Starting HiveServer2](#)
- [Using Beeline](#)

The upgrade is now complete.

## Installing Hive

Install the appropriate Hive packages using the appropriate command for your distribution.

RedHat and CentOS systems	<code>\$ sudo yum install &lt;pkg1&gt; &lt;pkg2&gt; ...</code>
SLES systems	<code>\$ sudo zypper install &lt;pkg1&gt; &lt;pkg2&gt; ...</code>
Ubuntu and Debian systems	<code>\$ sudo apt-get install &lt;pkg1&gt; &lt;pkg2&gt; ...</code>

The packages are:

- **hive** – base package that provides the complete language and runtime (required)
- **hive-metastore** – provides scripts for running the metastore as a standalone service (optional)
- **hive-server** – provides scripts for running the original HiveServer as a standalone service (optional)



- **hive-server2** – provides scripts for running the new HiveServer2 as a standalone service (optional)

## Configuring the Hive Metastore

The Hive metastore service stores the metadata for Hive tables and partitions in a relational database, and provides clients (including Hive) access to this information via the metastore service API. The subsections that follow discuss the deployment options and provide instructions for setting up a database in a recommended configuration.

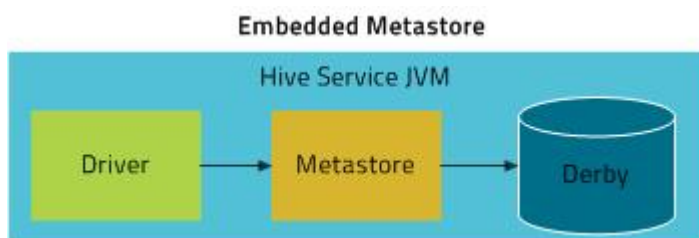
### Metastore Deployment Modes

- **Note:**

**HiveServer** in the discussion that follows refers to HiveServer1 or HiveServer2, whichever you are using.

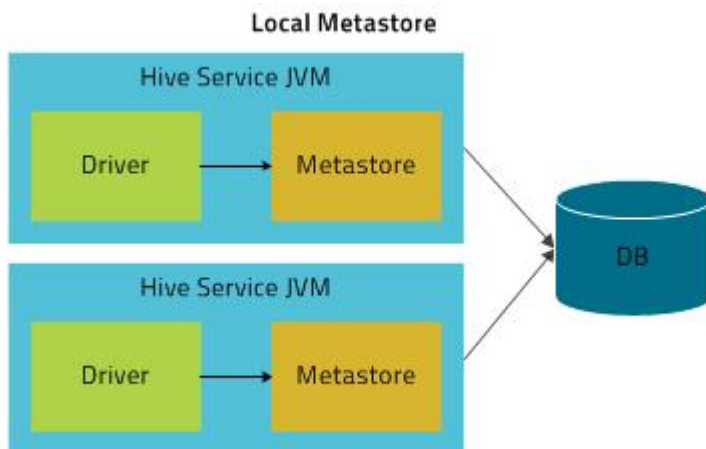
#### Embedded Mode

Cloudera recommends using this mode for experimental purposes only.



This is the default metastore deployment mode for CDH. In this mode the metastore uses a Derby database, and both the database and the metastore service run embedded in the main HiveServer process. Both are started for you when you start the HiveServer process. This mode requires the least amount of effort to configure, but it can support only one active user at a time and is not certified for production use.

#### Local Mode

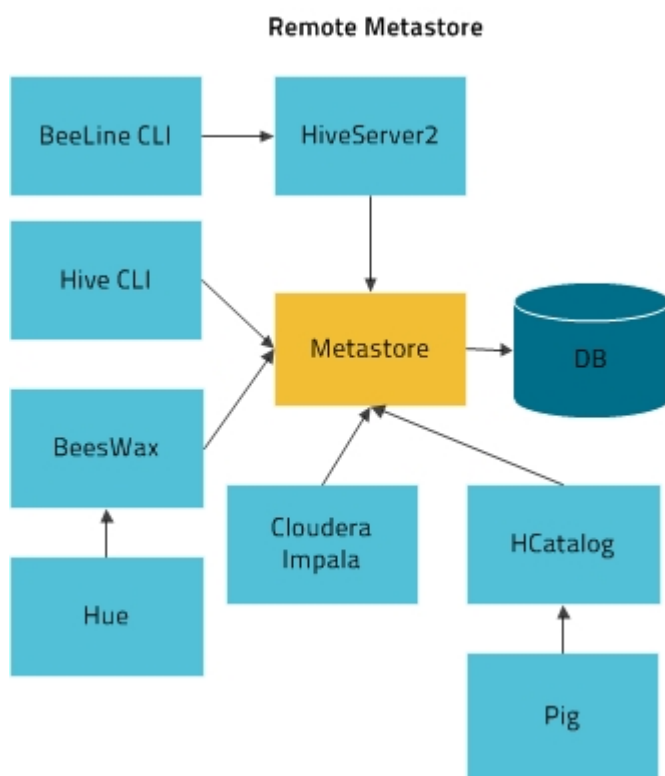


## Hive Installation

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

### Remote Mode

Cloudera recommends that you use this mode.



In this mode the Hive metastore service runs in its own JVM process; HiveServer2, HCatalog, Cloudera Impala™, and other processes communicate with it via the Thrift network API (configured via the `hive.metastore.uris` property). The metastore service communicates with the metastore database over JDBC (configured via the `javax.jdo.option.ConnectionURL` property). The database, the HiveServer process, and the metastore service can all be on the same host, but running the HiveServer process on a separate host provides better availability and scalability.

The main advantage of Remote mode over Local mode is that Remote mode does not require the administrator to share JDBC login information for the metastore database with each Hive user. [HCatalog](#) requires this mode.

## Supported Metastore Databases

See the [CDH4 Requirements and Supported Versions page](#) for up-to-date information on supported databases. Cloudera strongly encourages you to use MySQL because it is the most popular with the rest of the Hive user community, and so receives more testing than the other options.

## Configuring the Metastore Database

This section describes how to configure Hive to use a remote database, with examples for [MySQL](#) and [PostgreSQL](#).

The configuration properties for the Hive metastore are documented on the [Hive Metastore documentation](#) page, which also includes a pointer to the E/R diagram for the Hive metastore.

■ **Note:**

For information about additional configuration that may be needed in a secure cluster, see [Hive Security Configuration](#).

### Configuring a remote MySQL database for the Hive Metastore

Cloudera recommends you configure a database for the metastore on one or more remote servers (that is, on a host or hosts separate from the HiveServer1 or HiveServer2 process). MySQL is the most popular database to use. Proceed as follows.

#### Step 1: Install and start MySQL if you have not already done so

To install MySQL on a Red Hat system:

```
$ sudo yum install mysql-server
```

To install MySQL on a SLES system:

```
$ sudo zypper install mysql
$ sudo zypper install libmysqlclient_r15
```

To install MySQL on an Debian/Ubuntu system:

```
$ sudo apt-get install mysql-server
```

After using the command to install MySQL, you may need to respond to prompts to confirm that you do want to complete the installation. After installation completes, start the `mysql` daemon.

**On Red Hat systems**

```
$ sudo service mysqld start
```

**On SLES and Debian/Ubuntu systems**

```
$ sudo service mysql start
```

#### Step 2: Configure the MySQL Service and Connector

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

**To install the MySQL connector on a Red Hat 6 system:**

Install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo yum install mysql-connector-java
$ ln -s /usr/share/java/mysql-connector-java.jar
  /usr/lib/hive/lib/mysql-connector-java.jar
```

**To install the MySQL connector on a Red Hat 5 system:**

Download the MySQL JDBC driver from <http://www.mysql.com/downloads/connector/j/5.1.html>. You will need to sign up for an account if you don't already have one, and log in, before you can download it. Then copy it to the `/usr/lib/hive/lib/` directory. For example:

```
$ sudo cp mysql-connector-java-version/mysql-connector-java-version-bin.jar
  /usr/lib/hive/lib/
```

- **Note:** At the time of publication, *version* was 5.1.25, but the version may have changed by the time you read this.

### To install the MySQL connector on a SLES system:

Install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo zypper install mysql-connector-java
$ ln -s /usr/share/java/mysql-connector-java.jar
/usr/lib/hive/lib/mysql-connector-java.jar
```

### To install the MySQL connector on a Debian/Ubuntu system:

Install `mysql-connector-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

- **Note:** For Ubuntu Precise systems, the name of the connector JAR file is `mysql.jar`. Modify the second command given below accordingly.

```
$ sudo apt-get install libmysql-java
$ ln -s /usr/share/java/libmysql-java.jar /usr/lib/hive/lib/libmysql-java.jar
```

Configure MySQL to use a strong password and to start at boot. Note that in the following procedure, your current root password is blank. Press the Enter key when you're prompted for the root password.

### To set the MySQL root password:

```
$ sudo /usr/bin/mysql_secure_installation
[...]
Enter current password for root (enter for none):
OK, successfully used password, moving on...
[...]
Set root password? [Y/n] y
New password:
Re-enter new password:
Remove anonymous users? [Y/n] Y
[...]
Disallow root login remotely? [Y/n] N
[...]
Remove test database and access to it [Y/n] Y
[...]
Reload privilege tables now? [Y/n] Y
All done!
```

### To make sure the MySQL server starts at boot:

- On Red Hat systems:

```
$ sudo /sbin/chkconfig mysqld on
$ sudo /sbin/chkconfig --list mysqld
mysqld          0:off  1:off  2:on   3:on   4:on   5:on   6:off
```

- On SLES systems:

```
$ sudo chkconfig --add mysql
```

- On Debian/Ubuntu systems:

```
$ sudo chkconfig mysql on
```

## Step 3. Create the Database and User

The instructions in this section assume you are using [Remote mode](#), and that the MySQL database is installed on a separate host from the metastore service, which is running on a host named `metastorehost` in the example.

- Note:**

If the metastore service will run on the host where the database is installed, replace 'metastorehost' in the CREATE USER example with 'localhost'. Similarly, the value of `javax.jdo.option.ConnectionURL` in `/etc/hive/conf/hive-site.xml` (discussed in the [next step](#)) must be `jdbc:mysql://localhost/metastore`. For more information on adding MySQL users, see <http://dev.mysql.com/doc/refman/5.5/en/adding-users.html>.

Create the initial database schema using the `hive-schema-0.10.0.mysql.sql` file located in the `/usr/lib/hive/scripts/metastore/upgrade/mysql` directory.

### Example

```
$ mysql -u root -p
Enter password:
mysql> CREATE DATABASE metastore;
mysql> USE metastore;
mysql> SOURCE /usr/lib/hive/scripts/metastore/upgrade/mysql/hive-schema-0.10.0.mysql.sql;
```

You also need a MySQL user account for Hive to use to access the metastore. It is very important to prevent this user account from creating or altering tables in the metastore database schema.

- Important:**

If you fail to restrict the ability of the metastore MySQL user account to create and alter tables, it is possible that users will inadvertently corrupt the metastore schema when they use older or newer versions of Hive.

### Example

```
mysql> CREATE USER 'hive'@'metastorehost' IDENTIFIED BY 'mypassword';
...
mysql> REVOKE ALL PRIVILEGES, GRANT OPTION FROM 'hive'@'metastorehost';
mysql> GRANT SELECT,INSERT,UPDATE,DELETE,LOCK TABLES,EXECUTE ON metastore.* TO
'hive'@'metastorehost';
mysql> FLUSH PRIVILEGES;
mysql> quit;
```

## Step 4: Configure the Metastore Service to Communicate with the MySQL Database

This step shows the configuration properties you need to set in `hive-site.xml` to configure the metastore service to communicate with the MySQL database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

Given a MySQL database running on `myhost` and the user account `hive` with the password `mypassword`, set the configuration as follows (overwriting any existing values).

- Note:**

The `hive.metastore.local` property is no longer supported as of Hive 0.10; setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://myhost/metastore</value>
  <description>the URL of the MySQL database</description>
</property>

<property>
```

```
<name>javax.jdo.option.ConnectionDriverName</name>
<value>com.mysql.jdbc.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hive</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>

<property>
  <name>datanucleus.autoStartMechanism</name>
  <value>SchemaTable</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
  host</description>
</property>
```

### Configuring a remote PostgreSQL database for the Hive Metastore

Before you can run the Hive metastore with a remote PostgreSQL database, you must configure a connector to the remote PostgreSQL database, set up the initial database schema, and configure the PostgreSQL user account for the Hive user.

#### Step 1: Install and start PostgreSQL if you have not already done so

##### To install PostgreSQL on a Red Hat system:

```
$ sudo yum install postgresql-server
```

##### To install PostgreSQL on a SLES system:

```
$ sudo zypper install postgresql-server
```

##### To install PostgreSQL on an Debian/Ubuntu system:

```
$ sudo apt-get install postgresql
```

After using the command to install PostgreSQL, you may need to respond to prompts to confirm that you do want to complete the installation. In order to finish installation on Red Hat compatible systems, you need to initialize the database. Please note that this operation is not needed on Ubuntu and SLES systems as it's done automatically on first start:

##### To initialize database files on Red Hat compatible systems

```
$ sudo service postgresql initdb
```

To ensure that your PostgreSQL server will be accessible over the network, you need to do some additional configuration.

First you need to edit the `postgresql.conf` file. Set the `listen_addresses` property to `*`, to make sure that the PostgreSQL server starts listening on all your network interfaces. Also make sure that the `standard_conforming_strings` property is set to `off`.

You can check that you have the correct values as follows:

#### On Red-Hat-compatible systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

#### On SLES systems:

```
$ sudo cat /var/lib/pgsql/data/postgresql.conf | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

#### On Ubuntu and Debian systems:

```
$ cat /etc/postgresql/9.1/main/postgresql.conf | grep -e listen -e
standard_conforming_strings
listen_addresses = '*'
standard_conforming_strings = off
```

You also need to configure authentication for your network in `pg_hba.conf`. You need to make sure that the PostgreSQL user that you will create in the [next step](#) will have access to the server from a remote host. To do this, add a new line into `pg_hba.conf` that has the following information:

host	<database>	<user>	<network address>	<mask>
md5				

The following example allows all users to connect from all hosts to all your databases:

host	all	all	0.0.0.0	0.0.0.0	md5
------	-----	-----	---------	---------	-----

#### Note:

This configuration is applicable only for a network listener. Using this configuration won't open all your databases to the entire world; the user must still supply a password to authenticate himself, and privilege restrictions configured in PostgreSQL will still be applied.

After completing the installation and configuration, you can start the database server:

#### Start PostgreSQL Server

```
$ sudo service postgresql start
```

Use `chkconfig` utility to ensure that your PostgreSQL server will start at a boot time. For example:

```
chkconfig postgresql on
```

You can use the `chkconfig` utility to verify that PostgreSQL server will be started at boot time, for example:

```
chkconfig --list postgresql
```

### Step 2: Install the Postgres JDBC Driver

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

#### To install the PostgreSQL JDBC Driver on a Red Hat 6 system:

Install `postgresql-jdbc` package and create symbolic link to the `/usr/lib/hive/lib/` directory. For example:

```
$ sudo yum install postgresql-jdbc
$ ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-jdbc.jar
```

#### To install the PostgreSQL connector on a Red Hat 5 system:

You need to manually download the PostgreSQL connector from <http://jdbc.postgresql.org/download.html> and move it to the `/usr/lib/hive/lib/` directory. For example:

```
$ wget http://jdbc.postgresql.org/download/postgresql-9.2-1002.jdbc4.jar
$ mv postgresql-9.2-1002.jdbc4.jar /usr/lib/hive/lib/
```

#### Note:

You may need to use a different version if you have a different version of Postgres. You can check the version as follows:

```
$ sudo rpm -qa | grep postgres
```

#### To install the PostgreSQL JDBC Driver on a SLES system:

Install `postgresql-jdbc` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo zypper install postgresql-jdbc
$ ln -s /usr/share/java/postgresql-jdbc.jar /usr/lib/hive/lib/postgresql-jdbc.jar
```

#### To install the PostgreSQL JDBC Driver on a Debian/Ubuntu system:

Install `libpostgresql-jdbc-java` and symbolically link the file into the `/usr/lib/hive/lib/` directory.

```
$ sudo apt-get install libpostgresql-jdbc-java
$ ln -s /usr/share/java/postgresql-jdbc4.jar /usr/lib/hive/lib/postgresql-jdbc4.jar
```

### Step 3: Create the metastore database and user account

Proceed as in the following example:

```
$ sudo -u postgres psql
postgres=# CREATE USER hiveuser WITH PASSWORD 'mypassword';
postgres=# CREATE DATABASE metastore;
postgres=# \c metastore;
You are now connected to database 'metastore'.
postgres=# \i
/usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schema-0.10.0.postgres.sql
SET
SET
...
```

Now you need to grant permission for all metastore tables to user `hiveuser`. PostgreSQL does not have statements to grant the permissions for all tables at once; you'll need to grant the permissions one table at a time. You could automate the task with the following SQL script:



■ **Note:**

If you are running these commands interactively and are still in the Postgres session initiated at the beginning of this step, you do not need to repeat `sudo -u postgres psql`.

```
bash# sudo -u postgres psql
metastore=# \pset tuples_only on
metastore=# \o /tmp/grant-privs
metastore=# SELECT 'GRANT SELECT,INSERT,UPDATE,DELETE ON "' || schemaname || '".' ||
||tablename ||'" TO hiveuser ;'
metastore=# FROM pg_tables
metastore=# WHERE tableowner = CURRENT_USER and schemaname = ' public';
metastore=# \o
metastore=# \pset tuples_only off
metastore=# \i /tmp/grant-privs
```

#### Step 4: Configure the Metastore Service to Communicate with the PostgreSQL Database

This step shows the configuration properties you need to set in `hive-site.xml` to configure the metastore service to communicate with the PostgreSQL database. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that **must** be configured on all of them; the others are used only on the metastore host.

Given a PostgreSQL database running on host `myhost` under the user account `hive` with the password `mypassword`, you would set configuration properties as follows.

■ **Note:**

- The instructions in this section assume you are using [Remote mode](#), and that the PostgreSQL database is installed on a separate host from the metastore server.
- The `hive.metastore.local` property is no longer supported as of Hive 0.10; setting `hive.metastore.uris` is sufficient to indicate that you are using a remote metastore.

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:postgresql://myhost/metastore</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>org.postgresql.Driver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hiveuser</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>>false</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>
```

### Step 6: Test connectivity to the metastore:

```
$ hive -e "show tables;"
```

- **Note:**

This will take a while the first time.

### Configuring a remote Oracle database for the Hive Metastore

Before you can run the Hive metastore with a remote Oracle database, you must configure a connector to the remote Oracle database, set up the initial database schema, and configure the Oracle user account for the Hive user.

#### Step 1: Install and start Oracle

The Oracle database is not part of any Linux distribution and must be purchased, downloaded and installed separately. You can use the [Express edition](#), which can be downloaded free from Oracle website.

#### Step 2: Install the Oracle JDBC Driver

You must download the Oracle JDBC Driver from the Oracle website and put the file `ojdbc6.jar` into `/usr/lib/hive/lib/` directory. The driver is available for download [here](#).

- **Note:**

These URLs were correct at the time of publication, but the Oracle site is restructured frequently.

```
$ sudo mv ojdbc6.jar /usr/lib/hive/lib/
```

#### Step 3: Create the Metastore database and user account

Connect to your Oracle database as an administrator and create the user that will use the Hive metastore.

```
$ sqlplus "sys as sysdba"
SQL> create user hiveuser identified by mypassword;
SQL> grant connect to hiveuser;
SQL> grant all privileges to hiveuser;
```

Connect as the newly created `hiveuser` user and load the initial schema:

```
$ sqlplus hiveuser
SQL> @/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-0.10.0.oracle.sql
```

Connect back as an administrator and remove the power privileges from user `hiveuser`. Then grant limited access to all the tables:

```
$ sqlplus "sys as sysdba"
SQL> revoke all privileges from hiveuser;
SQL> BEGIN
2     FOR R IN (SELECT owner, table_name FROM all_tables WHERE owner='HIVEUSER') LOOP
3         EXECUTE IMMEDIATE 'grant SELECT,INSERT,UPDATE,DELETE on
'||R.owner||'.'||R.table_name||' to hiveuser';
4     END LOOP;
5 END;
6
7 /
```

#### Step 4: Configure the Metastore Service to Communicate with the Oracle Database

This step shows the configuration properties you need to set in `hive-site.xml` to configure the metastore service to communicate with the Oracle database, and provides sample settings. Though you can use the same `hive-site.xml` on all hosts (client, metastore, HiveServer), `hive.metastore.uris` is the only property that must be configured on all of them; the others are used only on the metastore host.

### Example

Given an Oracle database running on `myhost` and the user account `hiveuser` with the password `mypassword`, set the configuration as follows (overwriting any existing values):

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:oracle:thin:@//myhost/xe</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>oracle.jdbc.OracleDriver</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionUserName</name>
  <value>hiveuser</value>
</property>

<property>
  <name>javax.jdo.option.ConnectionPassword</name>
  <value>mypassword</value>
</property>

<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>>false</value>
</property>

<property>
  <name>datanucleus.fixedDatastore</name>
  <value>true</value>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and port of the metastore
host</description>
</property>
```

## Configuring HiveServer2

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

### Table Lock Manager (Required)

You must properly configure and enable Hive's Table Lock Manager. This requires installing ZooKeeper and setting up a ZooKeeper ensemble; see [ZooKeeper Installation](#).

- **Important:**

Failure to do this will prevent HiveServer2 from handling concurrent query requests and may result in data corruption.

## Hive Installation

Enable the lock manager by setting properties in `/etc/hive/conf/hive-site.xml` as follows (substitute your actual ZooKeeper node names for those in the example):

```
<property>
  <name>hive.support.concurrency</name>
  <description>Enable Hive's Table Lock Manager Service</description>
  <value>true</value>
</property>

<property>
  <name>hive.zookeeper.quorum</name>
  <description>Zookeeper quorum used by Hive's Table Lock Manager</description>
  <value>zk1.myco.com,zk2.myco.com,zk3.myco.com</value>
</property>
```

■

**Important:**  
Enabling the Table Lock Manager without specifying a list of valid Zookeeper quorum nodes will result in unpredictable behavior. Make sure that both properties are properly configured.

### hive.zookeeper.client.port

If ZooKeeper is not using the default value for ClientPort, you need to set `hive.zookeeper.client.port` in `/etc/hive/conf/hive-site.xml` to the same value that ZooKeeper is using. Check `/etc/zookeeper/conf/zoo.cfg` to find the value for ClientPort. If ClientPort is set to any value other than 2181 (the default), set `hive.zookeeper.client.port` to the same value. For example, if ClientPort is set to 2222, set `hive.zookeeper.client.port` to 2222 as well:

```
<property>
  <name>hive.zookeeper.client.port</name>
  <value>2222</value>
  <description>
    The port at which the clients will connect.
  </description>
</property>
```

### JDBC driver

The connection URL format and the driver class are different for HiveServer2 and HiveServer1:

HiveServer version	Connection URL	Driver Class
HiveServer2	<code>jdbc:hive2://&lt;host&gt;:&lt;port&gt;</code>	<code>org.apache.hive.jdbc.HiveDriver</code>
HiveServer1	<code>jdbc:hive://&lt;host&gt;:&lt;port&gt;</code>	<code>org.apache.hadoop.hive.jdbc.HiveDriver</code>

### Authentication

HiveServer2 can be [configured](#) to authenticate all connections; by default, it allows any client to connect. HiveServer2 supports either [Kerberos](#) or [LDAP](#) authentication; configure this in the `hive.server2.authentication` property in the `hive-site.xml` file. You can also configure [Pluggable Authentication](#), which allows you to use a custom authentication provider for HiveServer2; and [HiveServer2 Impersonation](#), which allows users to execute queries and access HDFS files as the connected user rather than the super user who started the HiveServer2 daemon. For more information, see [Hive Security Configuration](#).

In addition, for non-Kerberos connections, you can configure Secure Socket Layer (SSL) communication between HiveServer2 and clients. See [Configuring Encrypted Client/Server Communication for non-Kerberos HiveServer2 Connections](#)

## Configuring HiveServer2 for YARN

To use HiveServer2 with YARN, you must set the `HADOOP_MAPRED_HOME` environment variable: add the following line to `/etc/default/hive-server2`:

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

## Running HiveServer2 and HiveServer Concurrently

Cloudera recommends running HiveServer2 instead of the original HiveServer (HiveServer1) package in most cases; HiveServer1 is included for backward compatibility. Both HiveServer2 and HiveServer1 can be run concurrently on the same system, sharing the same data sets. This allows you to run HiveServer1 to support, for example, Perl or Python scripts that use the native HiveServer1 Thrift bindings.

Both HiveServer2 and HiveServer1 bind to port 10000 by default, so at least one of them must be configured to use a different port. You can set the port for HiveServer2 in `hive-site.xml` by means of the `hive.server2.thrift.port` property. For example:

```
<property>
  <name>hive.server2.thrift.port</name>
  <value>10001</value>
  <description>TCP port number to listen on, default 10000</description>
</property>
```

You can also specify the port (and the host IP address in the case of HiveServer2) by setting these environment variables:

HiveServer version	Port	Host Address
HiveServer2	<code>HIVE_SERVER2_THRIFT_PORT</code>	<code>HIVE_SERVER2_THRIFT_BIND_HOST</code>
HiveServer1	<code>HIVE_PORT</code>	<i>&lt;Host bindings cannot be specified&gt;</i>

## Using Custom UDFs with HiveServer2

To use custom User-Defined Functions (UDFs) with HiveServer2, do the following:

1. Copy the UDF JAR files to the machine(s) hosting the HiveServer2 server(s).  
Save the JARs to any directory you choose, and make a note of the path.
2. Make the JARs available to the current instance of HiveServer2 by setting `HIVE_AUX_JARS_PATH` to the JARs' full pathname (the one you noted in Step 1) in `hive-config.sh`

■ **Note:**

The path can be the directory, or each JAR's full pathname in a comma-separated list.

If you are using Cloudera Manager, use the HiveServer2 Service Environment Safety Valve to set `HIVE_AUX_JARS_PATH`.

3. Add each JAR file's full pathname to the `hive.aux.jars.path` config property in `hive-site.xml` and re-start HiveServer2.

This is to allow JARs to be passed to MapReduce jobs started by Hive.

## Starting the Metastore

- **Important:**

If you are running the metastore in [Remote mode](#), you **must** start the metastore before starting HiveServer2.

You can run the metastore from the command line:

```
$ hive --service metastore
```

Use Ctrl-c to stop the metastore process running from the command line.

To run the metastore as a daemon, the command is:

```
$ sudo service hive-metastore start
```

## File System Permissions

Your Hive data is stored in HDFS, normally under `/user/hive/warehouse` (or any path you specify as `hive.metastore.warehouse.dir` in your `hive-site.xml`). Make sure this location exists and is writable by the users whom you expect to be creating tables.

- **Important:**

Cloudera recommends setting permissions on the Hive warehouse directory to 1777, making it accessible to all users, with the sticky bit set. This allows users to create and access their tables, but prevents them from deleting tables they don't own.

In addition, each user submitting queries must have an HDFS home directory. `/tmp` (on the local file system) must be world-writable, as Hive makes extensive use of it.

[HiveServer2 Impersonation](#) allows users to execute queries and access HDFS files as the connected user.

If you do not enable impersonation, HiveServer2 by default executes all Hive tasks as the user ID that starts the Hive server; for clusters that use Kerberos authentication, this is the ID that maps to the [Kerberos principal](#) used with HiveServer2. Setting permissions to 1777, as recommended above, allows this user access to the Hive warehouse directory.

You can change this default behavior by setting `hive.metastore.execute.setugi` to `true` *on both the server and client*. This setting causes the metastore server to use the client's user and group permissions.

See also [Hive and HDFS Trash](#) on page 206.

## Hive and HDFS Trash

If the HDFS Trash function is [enabled](#), operations such as `ALTER TABLE DROP PARTITION` and `DROP TABLE` will move the data to the trash (`.Trash`) directory rather than removing it permanently. You will be able to restore the data from the trash if you need to.

- **Note:** Metadata is not saved in the trash and is removed permanently; you will not be able to restore the metadata of dropped tables, partitions, etc.

If the trash directory is not accessible the data will remain in the original location (in the Hive warehouse directory for internal tables).

Which trash directory is used depends on the way you are accessing Hive:

- If you use the `hive` command to drop the data, the data is moved to the trash directory in the home directory of the user executing the `hive` command.
- If you use HiveServer2:
  - If impersonation is enabled, the data is moved to the trash directory in the home directory of the user authorized to run HiveServer2.
  - If impersonation is disabled, the data is moved to the trash directory in the home directory of the user who is running HiveServer2. (HiveServer2 usually runs as user `hive`).

## Starting, Stopping, and Using HiveServer2

HiveServer2 is an improved version of HiveServer that supports Kerberos authentication and multi-client concurrency. Cloudera recommends HiveServer2.

▪ **Warning:**

If you are running the metastore in [Remote mode](#), you must start the Hive metastore before you start HiveServer2. HiveServer2 tries to communicate with the metastore as part of its initialization bootstrap. If it is unable to do this, it fails with an error.

### To start HiveServer2:

```
$ sudo service hive-server2 start
```

### To stop HiveServer2:

```
$ sudo service hive-server2 stop
```

To confirm that HiveServer2 is working, start the beeline CLI and use it to execute a `SHOW TABLES` query on the HiveServer2 process:

```
$ /usr/lib/hive/bin/beeline
beeline> !connect jdbc:hive2://localhost:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000> SHOW TABLES;
show tables;
+-----+
| tab_name |
+-----+
No rows selected (0.238 seconds)
0: jdbc:hive2://localhost:10000>
```

## Using the Beeline CLI

Beeline is a new CLI (command-line interface) for HiveServer2. It is based on the [SQLLine CLI](#) written by Marc Prud'hommeaux.

- **Note:** You cannot use Beeline to communicate with the original HiveServer (HiveServer1).

## Hive Installation

Use the following commands to start `beeline` and connect to a running `HiveServer2` process. In this example the `HiveServer2` process is running on `localhost` at port `10000`:

```
$ beeline
beeline> !connect jdbc:hive2://localhost:10000 username password
org.apache.hive.jdbc.HiveDriver
0: jdbc:hive2://localhost:10000>
```

- **Note:**

If you are using `HiveServer2` on a cluster that does *not* have Kerberos security enabled, then the password is arbitrary in the command for starting Beeline.

If you are using `HiveServer2` on a cluster that does have Kerberos security enabled, please refer to the [CDH 4 Security Guide](#).

At present the best source for documentation on Beeline is the original [SQLLine documentation](#).

## Starting HiveServer1 and the Hive Console

- **Note:** The Hive Console is not needed if you are using [Beeline](#) with `HiveServer2`.

To start `HiveServer1`:

```
$ sudo service hiveserver start
```

See also [Running HiveServer2 and HiveServer Concurrently](#) on page 205

To start the Hive console:

```
$ hive
hive>
```

To confirm that Hive is working, issue the `show tables;` command to list the Hive tables; be sure to use a semi-colon after the command:

```
hive> show tables;
OK
Time taken: 10.345 seconds
```

## Using Hive with HBase

To allow Hive scripts to use HBase, add the following statements to the top of each script. Replace the `<component_version>` strings with current version numbers for CDH, Guava and the Hive HBase handler. (You can find current version numbers for CDH dependencies such as Guava in CDH's `root.pom.xml` file for the current release, for example [cdh-root-4.4.0.pom](#).)

```
ADD JAR /usr/lib/hive/lib/zookeeper.jar
ADD JAR /usr/lib/hive/lib/hbase.jar
ADD JAR
/usr/lib/hive/lib/hive-hbase-handler-<Hive-HBase-Handler_version>-cdh<CDH_version>.jar
ADD JAR /usr/lib/hive/lib/guava-<Guava_version>.jar
```



For example,

```
ADD JAR /usr/lib/hive/lib/zookeeper.jar
ADD JAR /usr/lib/hive/lib/hive-hbase-handler-0.10.0-cdh4.4.0.jar
ADD JAR /usr/lib/hive/lib/guava-11.0.2.jar
```

■ **Note:**

Instead of adding these statements to each script, you can populate the `hive.aux.jars.path` property in `hive-site.xml`; for example:

```
<property>
  <name>hive.aux.jars.path </name>
  <value>file:///usr/lib/hive/lib/zookeeper.jar;file:///usr/lib/hive/lib/hive-hbase-handler-0.10.0-cdh4.4.0.jar;file:///usr/lib/hive/lib/guava-11.0.2.jar;file:///usr/lib/hive/lib/hive.jar</value>
</property>
```

## Installing the Hive JDBC on Clients

If you want to install only the JDBC on your Hive clients, proceed as follows.

1. Install the package (it is included in CDH packaging). Use one of the following commands, depending on the target operating system:

- On Red-Hat-compatible systems:

```
$ sudo yum install hive-jdbc
```

- On SLES systems:

```
$ sudo zypper install hive-jdbc
```

- On Ubuntu or Debian systems:

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

2. Add `/usr/lib/hive/lib/*.jar` and `/usr/lib/hadoop/*.jar` to your classpath.

You are now ready to run your JDBC client. For more information see the [Hive Client](#) document.

## Setting HADOOP\_MAPRED\_HOME for YARN

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

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

## Configuring the Metastore to use HDFS High Availability

See [Upgrading the Hive Metastore to use HDFS HA](#).

## Troubleshooting

This section provides guidance on problems you may encounter while installing, upgrading, or running Hive.

### Hive Queries Fail with "Too many counters" Error

#### Explanation

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

■ **Note:**

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

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

#### What To Do

If you run into this error, set `mapreduce.job.counters.max` in `mapred-site.xml` to a higher value.

## Viewing the Hive Documentation

For additional Hive documentation, see [the Apache Hive wiki](#).

To view Cloudera's video tutorial about using Hive, see [Introduction to Apache Hive](#).

## Installing and Using HCatalog

Apache HCatalog provides table data access for CDH components such as Pig, Sqoop, and MapReduce. Table definitions are maintained in the Hive metastore, which HCatalog requires. HCatalog makes the same table information available to Hive, Pig, MapReduce, and REST clients. This page explains how to install and configure it for REST access and for MapReduce and Pig access. For Sqoop, see the [section on Sqoop-HCatalog integration](#) in the Sqoop User Guide.

Use the sections that follow to install, configure and use HCatalog:

- [Prerequisites](#)
- [Installing the Packages](#)
- [Host Configuration Changes](#)
- [Starting and Stopping the WebHCat REST Server](#)
- [Accessing Table Data with the Command-line API](#)
- [Accessing Table Data with MapReduce](#)
- [Accessing Table Data with Pig](#)
- [Accessing Table Data with REST](#)
- [Apache HCatalog Documentation](#)

For more information, see the [HCatalog documentation](#).

## HCatalog Prerequisites

- [An operating system supported by CDH4](#)
- [Oracle JDK](#)
- The Hive [metastore and its database](#). The Hive metastore must be running in [remote mode](#) (as a service).

## Installing the HCatalog RPM or Debian Packages

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

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

HCatalog comprises the following packages:

- `hcatalog` - HCatalog wrapper for accessing the Hive metastore, libraries for MapReduce and Pig, and a command-line program
- `webhcat` - A REST API server for HCatalog
- `webhcat-server` - Installs webhcat and a server init script

### Important:

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands to install HCatalog. For instructions, see [CDH4 Installation](#).

### Installing the WebHCat REST Server

**To install the WebHCat REST server on a Red Hat system:**

```
$ sudo yum install webhcat-server
```

**To install the WebHCat REST server components on an Ubuntu or other Debian system:**

```
$ sudo apt-get install webhcat-server
```

**To install the WebHCat REST server components on a SLES system:**

```
$ sudo zypper install webhcat-server
```

■ **Note:**

- It is not necessary to install WebHCat if you will not be using the REST API. Pig and MapReduce do not need it.
- You can change the default port 50111 by creating or editing the following file and restarting WebHCat:

```
/etc/webhcat/conf/webhcat-site.xml
```

The property to change is:

```
<configuration>
  <property>
    <name>templeton.port</name>
    <value>50111</value>
    <description>The HTTP port for the main server.</description>
  </property>
</configuration>
```

- To uninstall WebHCat you must remove two packages: `webhcat-server` and `webhcat`.

### Installing HCatalog for Use with Pig and MapReduce

On hosts that will be used to launch Pig scripts or MapReduce applications using table information, install HCatalog as follows: **To install the HCatalog client components on a Red Hat system:**

```
$ sudo yum install hcatalog
```

**To install the HCatalog client components on an Ubuntu or other Debian system:**

```
$ sudo apt-get install hcatalog
```

**To install the HCatalog client components on a SLES system:**

```
$ sudo zypper install hcatalog
```

## Configuration Change on Hosts Used with HCatalog

You must update `/etc/hive/conf/hive-site.xml` on all hosts where WebHCat will run, as well as all hosts where Pig or MapReduce will be used with HCatalog, so that Metastore clients know where to find the Metastore.

Add or edit the `hive.metastore.uris` property as follows:

```
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<hostname>:9083</value>
</property>
```

where `<hostname>` is the host where the HCatalog server components are running, for example `hive.examples.com`.

## Starting and Stopping the WebHCat REST server

```
$ sudo service webhcat-server start
$ sudo service webhcat-server stop
```

## Accessing Table Information with the HCatalog Command-line API

```
# Create a table
$ hcat -e "create table groups(name string,placeholder string,id int) row format
delimited fields terminated by ':' stored as textfile"
OK

# Get the schema for a table
$ hcat -e "desc groups"
OK
name string
placeholder string
id int

# Create another table
$ hcat -e "create table groupids(name string,id int)"
OK
```

See the [HCatalog documentation](#) for information on using the HCatalog command-line application.

## Accessing Table Data with MapReduce

The following example MapReduce program reads from the `groups` table (consisting of data from `/etc/group`), extracts the first and third columns, and inserts them into the `groupids` table.

```
package com.cloudera.test;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.conf.*;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapreduce.*;
import org.apache.hadoop.util.*;
import org.apache.hcatalog.common.*;
import org.apache.hcatalog.mapreduce.*;
import org.apache.hcatalog.data.*;
import org.apache.hcatalog.data.schema.*;

public class UseHCat extends Configured implements Tool {
    public static class Map extends Mapper<WritableComparable, HCatRecord, Text,
    IntWritable> {
        String groupname;
```

```

        @Override
        protected void map( WritableComparable key,
                           HCatRecord value,
                           org.apache.hadoop.mapreduce.Mapper<WritableComparable,
HCatRecord,
                           Text, IntWritable>.Context context)
            throws IOException, InterruptedException {
            // The group table from /etc/group has name, 'x', id
            groupname = (String) value.get(0);
            int id = (Integer) value.get(2);
            // Just select and emit the name and ID
            context.write(new Text(groupname), new IntWritable(id));
        }
    }

    public static class Reduce extends Reducer<Text, IntWritable,
                                           WritableComparable, HCatRecord> {

        protected void reduce( Text key,
                               java.lang.Iterable<IntWritable> values,
                               org.apache.hadoop.mapreduce.Reducer<Text, IntWritable,
                               WritableComparable, HCatRecord>.Context context)
            throws IOException, InterruptedException {
            // Only expecting one ID per group name
            Iterator<IntWritable> iter = values.iterator();
            IntWritable iw = iter.next();
            int id = iw.get();
            // Emit the group name and ID as a record
            HCatRecord record = new DefaultHCatRecord(2);
            record.set(0, key.toString());
            record.set(1, id);
            context.write(null, record);
        }
    }

    public int run(String[] args) throws Exception {
        Configuration conf = getConf();
        args = new GenericOptionsParser(conf, args).getRemainingArgs();

        // Get the input and output table names as arguments
        String inputTableName = args[0];
        String outputTableName = args[1];
        // Assume the default database
        String dbName = null;

        Job job = new Job(conf, "UseHCat");
        HCatInputFormat.setInput(job, InputJobInfo.create(dbName,
            inputTableName, null));
        job.setJarByClass(UseHCat.class);
        job.setMapperClass(Map.class);
        job.setReducerClass(Reduce.class);

        // An HCatalog record as input
        job.setInputFormatClass(HCatInputFormat.class);

        // Mapper emits a string as key and an integer as value
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);

        // Ignore the key for the reducer output; emitting an HCatalog record as value

        job.setOutputKeyClass(WritableComparable.class);
        job.setOutputValueClass(DefaultHCatRecord.class);
        job.setOutputFormatClass(HCatOutputFormat.class);

        HCatOutputFormat.setOutput(job, OutputJobInfo.create(dbName,
            outputTableName, null));
        HCatSchema s = HCatOutputFormat.getTableSchema(job);
        System.err.println("INFO: output schema explicitly set for writing:" + s);
        HCatOutputFormat.setSchema(job, s);
        return (job.waitForCompletion(true) ? 0 : 1);
    }
}

```

```
public static void main(String[] args) throws Exception {
    int exitCode = ToolRunner.run(new UseHCat(), args);
    System.exit(exitCode);
}
```

Load data from the local file system into the groups table:

```
$ hive -e "load data local inpath '/etc/group' overwrite into table groups"
```

After compiling and creating a JAR file, set up the environment that is needed for copying required JAR files to HDFS and then run the job, for example:

- **Note:** You can find current version numbers for CDH dependencies in CDH's root `pom.xml` file for the current release, for example [cdh-root-4.4.0.pom](#).)

```
$ export HCAT_HOME=/usr/lib/hcatalog
$ export HIVE_HOME=/usr/lib/hive
$ HCATJAR=$HCAT_HOME/share/hcatalog/hcatalog-core-0.5.0-cdh4.4.0.jar
$ HCATPIGJAR=$HCAT_HOME/share/hcatalog/hcatalog-pig-adapter-0.5.0-cdh4.4.0.jar
$ HIVE_VERSION=0.10.0-cdh4.4.0
$ export
$ LIBJARS=`echo $HADOOP_CLASSPATH | sed -e 's/://,/g'`
$ export LIBJARS=$LIBJARS,$HIVE_HOME/lib/antlr-runtime-3.4.jar

$ hadoop jar target/UseHCat-1.0.jar com.cloudera.test.UseHCat -files $HCATJAR -libjars
$LIBJARS groups groupids
```

## Accessing Table Data with Pig

When using table information from the Hive metastore with Pig, add the `-useHCatalog` option when invoking pig:

```
$ pig -useHCatalog test.pig
```

In the script, use HCatLoader to have table schema retrieved automatically:

```
A = LOAD 'groups' USING org.apache.hcatalog.pig.HCatLoader();
DESCRIBE A;
```

Output:

```
A: {name: chararray,placeholder: chararray,id: int}
```

## Accessing Table Information with REST

Table information can be retrieved from any host that has HTTP access to the host where the WebHCat server is running. A Web browser or an HTTP client such as curl or wget can be used to verify the functionality.

The base URL for REST access to table information is `http://<SERVERHOST>:50111/template/v1/ddl`.

Examples of specific URLs:

```
http://<SERVERHOST>:50111/templeton/v1/ddl/database/?user.name=hive
http://<SERVERHOST>:50111/templeton/v1/ddl/database/default/table/?user.name=hive
http://<SERVERHOST>:50111/templeton/v1/ddl/database/default/table/groups?user.name=hive
```

## Installing and Using HCatalog

Example output:

```
{"columns":[{"name":"name","type":"string"}, {"name":"placeholder","type":"string"}, {"name":"id","type":"int"}], "database":"default", "table":"groptable"}
```

## Viewing the HCatalog Documentation

See the [Apache HCatalog wiki](#).



# HBase Installation

Apache HBase provides large-scale tabular storage for Hadoop using the Hadoop Distributed File System (HDFS). Cloudera recommends installing HBase in a standalone mode before you try to run it on a whole cluster.

- **Note: Install Cloudera Repository**

Before using the instructions on this page to install or upgrade, install the Cloudera `yum`, `zypper`/`YaST` or `apt` repository, and install or upgrade CDH4 and make sure it is functioning correctly. For instructions, see [CDH4 Installation](#) and the instructions for [upgrading to CDH4](#) or [upgrading from an earlier CDH4 release](#).

- **Note: Running Services**

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

Use the following sections to install, update, and configure HBase:

- [Upgrading HBase](#)
- [Installing HBase](#)
- [Configuration Settings](#)
- [Starting HBase in Standalone Mode](#)
- [Configuring HBase in Pseudo-Distributed Mode](#)
- [Deploying HBase in a Cluster](#)
- [Using the Hbase Shell](#)
- [Using MapReduce with HBase](#)
- [Troubleshooting](#)
- [Apache HBase Documentation](#)
- [HBase Replication](#)
- [Configuring Snapshots](#)

## Upgrading HBase

- **Note:**

To see which version of HBase is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

- **Important:**

Check the [Known Issues and Work Arounds in CDH4](#) and [Incompatible Changes](#) for HBase before proceeding.

### About Checksums in CDH4.2

When you upgrade to CDH4.2, checksums are turned off by default. This is because of [incompatibilities](#) between the Hfile format used in HBase 0.92 (delivered in CDH4.1.x) and the format used in 0.94, which is delivered in CDH4.2. If you want to use checksums in CDH4.2, proceed as follows:

- **Warning:**

Once you turn on HBase checksums in CDH4.2, you will not be able to roll back to an earlier HBase version.

1. Upgrade to 4.2
2. Restart HBase
3. Turn on checksums by setting `hbase.regionserver.checksum.verify` to `true` in `hbase-site.xml`
4. Restart HBase

### Upgrading HBase from CDH3 to CDH4

To upgrade HBase from CDH3 to CDH4 proceed as follows.

- **Note:**

If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of HBase.

#### Step 1: Perform a Graceful Cluster Shutdown and Remove HBase

**To shut down the CDH3 version of HBase gracefully:**

1. Stop the Thrift server and clients, then stop the cluster.

- a. Stop the Thrift server and clients:

```
sudo service hadoop-hbase-thrift stop
```

- b. Stop the cluster by shutting down the master and the region servers:

- a. Use the following command on the master node:

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

- b. Use the following command on each node hosting a region server:

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

2. 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
```

or (if the CDH4 version of ZooKeeper is already running)

```
$ sudo /sbin/service zookeeper-server stop
```

3. It is a good idea to back up the `/hbaseznnode` before proceeding. By default, this is in `/var/zookeeper`.
4. If you have not already done so, remove the CDH3 version of ZooKeeper. See [Upgrading ZooKeeper from CDH3 to CDH4](#).
5. Remove HBase:

**To remove HBase on Red-Hat-compatible systems:**

```
$ sudo yum remove hadoop-hbase
```

**To remove HBase on SLES systems:**

```
$ sudo zypper remove hadoop-hbase
```

**To remove HBase on Ubuntu and Debian systems:**

```
$ sudo apt-get purge hadoop-hbase
```

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, **DO NOT PROCEED** before backing them up.

## Step 2: Install the new version of HBase

Follow directions under [Installing HBase](#).

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

### Step 3: Upgrade the LZO Plugin

- **Note:** Skip this step if you were not using the LZO plugin in CDH3.

You need to make sure that the proper libraries (native and .jar) are on the classpath and the library path; otherwise your CDH3 LZO files will not be readable. Do this after installing the new version of HBase, but before starting the HBase Master or Region Servers. Proceed as follows.

1. Add the LZO JAR file from `/usr/lib/hadoop-0.20/lib/` to the HBase classpath:

```
echo 'export
HBASE_CLASSPATH=$HBASE_CLASSPATH:/usr/lib/hadoop-0.20/lib/hadoop-lzo-20101122174751.20101122171345.552b3f9.jar'>
/etc/hbase/conf/hbase-env.sh
```

2. Copy the native libraries from the old location to the default location specified in `java.library.path`. For example, on an Red Hat 6 or CentOS 6 system:

```
cp /usr/lib/hadoop-0.20/lib/native/Linux-amd64-64/* /usr/lib/hadoop/lib/native/
```

- **Note:** The old location may be different on SLES, Ubuntu, or Debian systems.

## Upgrading HBase from an Earlier CDH4 Release

To upgrade HBase from an earlier CDH4 release, proceed as follows.

The instructions that follow assume that you are upgrading HBase as part of an upgrade to the latest CDH4 release, and have already performed the steps under [Upgrading from an Earlier CDH4 Release](#).

### Step 1: Perform a Graceful Cluster Shutdown

**To shut HBase down gracefully:**

1. Stop the Thrift server and clients, then stop the cluster.

- a. Stop the Thrift server and clients:

```
sudo service hbase-thrift stop
```

- b. Stop the cluster by shutting down the master and the region servers:

- a. Use the following command on the master node:

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

- b. Use the following command on each node hosting a region server:

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

2. Stop the ZooKeeper Server:

```
$ sudo service zookeeper-server stop
```

## Step 2: Install the new version of HBase

- Note:**

You may want to take this opportunity to upgrade ZooKeeper, but you do not *have* to upgrade Zookeeper before upgrading HBase; the new version of HBase will run with the older version of Zookeeper. For instructions on upgrading ZooKeeper, see [Upgrading ZooKeeper to CDH4](#).

It is a good idea to back up the `/hbase znode` before proceeding. By default, this is in `/var/lib/zookeeper`.

To install the new version of HBase, follow directions in the next section, [Installing HBase](#).

- Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

## Installing HBase

To install HBase On Red Hat-compatible systems:

```
$ sudo yum install hbase
```

To install HBase on Ubuntu and Debian systems:

```
$ sudo apt-get install hbase
```

To install HBase on SLES systems:

```
$ sudo zypper install hbase
```

- Note:** See also [Starting HBase in Standalone Mode](#) on page 223, [Configuring HBase in Pseudo-Distributed Mode](#) on page 225, and [Deploying HBase in a Distributed Cluster](#) on page 227.

To list the installed files on Ubuntu and Debian systems:

```
$ dpkg -L hbase
```

To list the installed files on Red Hat and SLES systems:

```
$ rpm -ql hbase
```

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

You are now ready to enable the server daemons you want to use with Hadoop. You can also enable Java-based client access by adding the JAR files in `/usr/lib/hbase/` and `/usr/lib/hbase/lib/` to your Java class path.

## Configuration Settings for HBase

This section contains information on configuring the Linux host and HDFS for HBase.

### Using DNS with HBase

HBase uses the local hostname to report its IP address. Both forward and reverse DNS resolving should work. If your 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. For more information about NTP, see the [NTP site](#).

### 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-like systems is insufficient. Any significant amount of loading will result in failures 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 [Apache HBase Book](#). You may also notice errors such as:

```
2010-04-06 03:04:37,542 INFO org.apache.hadoop.hdfs.DFSClient: Exception
increaseBlockOutputStream 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:

```
hdfs -      nofile 32768
hbase -     nofile 32768
```

■ **Note:**

- Only the root user can edit this file.
- If this change does not take effect, check other configuration files in the `/etc/security/limits.d` directory for lines containing the `hdfs` or `hbase` user and the `nofile` value. Such entries may be overriding the entries in `/etc/security/limits.conf`.

To apply the changes in `/etc/security/limits.conf` on Ubuntu and Debian systems, add the following line in the `/etc/pam.d/common-session` file:

```
session required pam_limits.so
```

## 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 (by default found in `/etc/hadoop/conf/hdfs-site.xml`) 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.DFSCClient: Could not obtain block
blk_XXXXXXXXXXXXXXXXXXXXXXXXX_YYYYYYY 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

- **Note:**

You can skip this section if you are already running HBase in distributed or pseudo-distributed mode.

By default, HBase ships configured for *standalone mode*. In this mode of operation, a single JVM hosts the HBase Master, an HBase Region Server, and a ZooKeeper quorum peer. In order to run HBase in standalone mode, you must install the HBase Master package:

## Installing the HBase Master

**To install the HBase Master on Red Hat-compatible systems:**

```
$ sudo yum install hbase-master
```

**To install the HBase Master on Ubuntu and Debian systems:**

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

**To install the HBase Master on SLES systems:**

```
$ sudo zypper install hbase-master
```

## HBase Installation

### Starting the HBase Master

- On Red Hat and SLES systems (using `.rpm` packages) you can now start the HBase Master by using the included service script:

```
$ sudo service hbase-master start
```

- On Ubuntu systems (using Debian packages) the HBase Master starts when the HBase package is installed.

To verify that the standalone installation is operational, visit `http://localhost:60010`. The list of Region Servers at the bottom of the page should include one entry for your local machine.

■ **Note:**

Although you have 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 `zookeeper-server`) or uninstall the `hadoop-zookeeper-server` (or `zookeeper`) package.

See also [Accessing HBase by using the HBase Shell](#), [Using MapReduce with HBase](#) and [Troubleshooting](#).

### Installing and Configuring REST

**To install REST on Red Hat-compatible systems:**

```
$ sudo yum install hbase-rest
```

**To install REST on Ubuntu and Debian systems:**

```
$ sudo apt-get install hbase-rest
```

**To install REST on SLES systems:**

```
$ sudo zypper install hbase-rest
```

You can use the `service` command to run an `init.d` script, `/etc/init.d/hbase-rest`, to start the REST server; for example:

```
$ sudo service hbase-rest start
```

The script starts the server by default on port 8080. This is a commonly used port and so may conflict with other applications running on the same host.

If you need change the port for the REST server, configure it in `hbase-site.xml`, for example:

```
<property>
  <name>hbase.rest.port</name>
  <value>60050</value>
</property>
```



- **Note:**

You can use `HBASE_REST_OPTS` in `hbase-env.sh` to pass other settings (such as heap size and GC parameters) to the REST server JVM.

## Configuring HBase in Pseudo-Distributed Mode

- **Note:** You can skip this section if you are already running HBase in distributed mode, or if you intend to use only standalone mode.

*Pseudo-distributed* mode differs from *standalone* mode in that each of the component processes run in a separate JVM.

- **Note: Before you start**

- This section assumes you have already installed the [HBase master](#) and gone through the [standalone](#) configuration steps.
- If the HBase master is already running in standalone mode, stop it as follows before continuing with pseudo-distributed configuration:
  - To stop the CDH3 version: `sudo service hadoop-hbase-master stop`, *or*
  - To stop the CDH4 version if that version is already running: `sudo service hbase-master stop`

## Modifying the HBase Configuration

To enable pseudo-distributed mode, you must first make some configuration changes. Open `/etc/hbase/conf/hbase-site.xml` in your editor of choice, and insert the following XML properties between the `<configuration>` and `</configuration>` tags. Be sure to replace `myhost` with the hostname of your HDFS NameNode (as specified by `fs.default.name` or `fs.defaultFS` in your `conf/core-site.xml` file); you may also need to change the port number from the default (8020).

```
<property>
  <name>hbase.cluster.distributed</name>
  <value>true</value>
</property>
<property>
  <name>hbase.rootdir</name>
  <value>hdfs://myhost:8020/hbase</value>
</property>
```

## Creating the /hbase Directory in HDFS

Before starting the HBase Master, you need to create the `/hbase` directory in HDFS. The HBase master runs as `hbase:hbase` so it does not have the required permissions to create a top level directory.

**To create the `/hbase` directory in HDFS:**

```
$ sudo -u hdfs hadoop fs -mkdir /hbase
$ sudo -u hdfs hadoop fs -chown hbase /hbase
```

- **Note:** If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) *or* `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

### Enabling Servers for Pseudo-distributed Operation

After you have configured HBase, you must enable the various servers that make up a distributed HBase cluster. HBase uses three required types of servers:

- [Installing and Starting ZooKeeper Server](#)
- [Starting the HBase Master](#)
- [Starting an HBase RegionServer](#)

#### Installing and Starting ZooKeeper Server

HBase uses ZooKeeper Server as a highly available, central location for cluster management. For example, it allows clients to locate the servers, and ensures that only one master is active at a time. For a small cluster, running a ZooKeeper node collocated with the NameNode is recommended. For larger clusters, contact Cloudera Support for configuration help.

Install and start the ZooKeeper Server in standalone mode by running the commands shown in the [Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server](#) on page 245

#### Starting the HBase Master

After ZooKeeper is running, you can start the HBase master in standalone mode.

```
$ sudo service hbase-master start
```

#### Starting an HBase RegionServer

The RegionServer is the part of HBase that actually hosts data and processes requests. The region server typically runs on all of the slave nodes in a cluster, but not the master node.

**To enable the HBase RegionServer On Red Hat-compatible systems:**

```
$ sudo yum install hbase-regionserver
```

**To enable the HBase RegionServer on Ubuntu and Debian systems:**

```
$ sudo apt-get install hbase-regionserver
```

**To enable the HBase RegionServer on SLES systems:**

```
$ sudo zypper install hbase-regionserver
```

**To start the RegionServer:**

```
$ sudo service hbase-regionserver start
```

#### Verifying the Pseudo-Distributed Operation

After you have started ZooKeeper, the Master, and a RegionServer, the pseudo-distributed cluster should be up and running. You can verify that each of the daemons is running using the `jps` tool from the Oracle JDK, which you can obtain from [here](#). If you are running a pseudo-distributed HDFS installation and a pseudo-distributed HBase installation on one machine, `jps` will show the following output:

```
$ sudo jps
32694 Jps
30674 HRegionServer
29496 HMaster
28781 DataNode
28422 NameNode
30348 QuorumPeerMain
```

You should also be able to navigate to `http://localhost:60010` and verify that the local region server has registered with the master.

## Installing and Starting the HBase Thrift Server

The HBase Thrift Server is an alternative gateway for accessing the HBase server. Thrift mirrors most of the HBase client APIs while enabling popular programming languages to interact with HBase. The Thrift Server is multiplatform and more performant than REST in many situations. Thrift can be run collocated along with the region servers, but should not be collocated with the NameNode or the JobTracker. For more information about Thrift, visit <http://incubator.apache.org/thrift/>.

**To enable the HBase Thrift Server On Red Hat-compatible systems:**

```
$ sudo yum install hbase-thrift
```

**To enable the HBase Thrift Server on Ubuntu and Debian systems:**

```
$ sudo apt-get install hbase-thrift
```

**To enable the HBase Thrift Server on SLES systems:**

```
$ sudo zypper install hbase-thrift
```

**To start the Thrift server:**

```
$ sudo service hbase-thrift start
```

See also [Accessing HBase by using the HBase Shell](#) on page 228, [Using MapReduce with HBase](#) on page 228 and [Troubleshooting](#) on page 229.

## 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.

- **Note: Before you start**

This section assumes that you have already installed the [Installing the HBase Master](#) on page 223 and the [HBase Region Server](#) and gone through steps for [standalone](#) and [pseudo-distributed](#) configuration. You are now about to distribute the processes across multiple hosts; see [Choosing Where to Deploy the Processes](#) on page 227.

## Choosing Where to Deploy the Processes

For small clusters, Cloudera recommends designating one node in your cluster as the master node. On this node, you will typically run the HBase Master and a ZooKeeper quorum peer. These master processes may be collocated with the Hadoop NameNode and JobTracker for small clusters.

Designate the remaining nodes as slave nodes. On each node, Cloudera recommends running a Region Server, which may be collocated with a Hadoop TaskTracker (MRv1) and a DataNode. When co-locating with TaskTrackers, be sure that the resources of the machine are not oversubscribed – it's safest to start with a small number of MapReduce slots and work up slowly.

### Configuring for Distributed Operation

After you have decided which machines will run each process, you can edit the configuration so that the nodes 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 in `hbase-site.xml`. Insert the following XML property to configure the nodes with the address of the node where the ZooKeeper quorum peer is running:

```
<property>
  <name>hbase.zookeeper.quorum</name>
  <value>mymasternode</value>
</property>
```

The `hbase.zookeeper.quorum` property is a comma-separated list of hosts on which ZooKeeper servers are running. If one of the ZooKeeper servers is down, HBase will use another from the list. By default, the ZooKeeper service is bound to port 2181. To change the port, add the `hbase.zookeeper.property.clientPort` property to `hbase-site.xml` and set the value to the port you want ZooKeeper to use. For more information, see [this chapter](#) of the Apache HBase Reference Guide.

To start the cluster, start the services in the following order:

1. The ZooKeeper Quorum Peer
2. The HBase Master
3. Each of the HBase RegionServers

After the cluster is fully started, you can view the HBase Master web interface on port 60010 and verify that each of the slave nodes has registered properly with the master.

See also [Accessing HBase by using the HBase Shell](#) on page 228, [Using MapReduce with HBase](#) on page 228 and [Troubleshooting](#) on page 229. For instructions on improving the performance of local reads, see [Tips and Guidelines](#) on page 107.

### Accessing HBase by using the HBase Shell

After you have started HBase, you can access the database by using the HBase Shell:

```
$ hbase shell
```

### Using MapReduce with HBase

To run MapReduce jobs that use HBase, you need to add the HBase and Zookeeper JAR files to the Hadoop Java classpath. You can do this by adding the following statement to each job:

```
TableMapReduceUtil.addDependencyJars(job);
```

This distributes the JAR files to the cluster along with your job and adds them to the job's classpath, so that you do not need to edit the MapReduce configuration.

You can find more information about `addDependencyJars` in the documentation listed under [Viewing the HBase Documentation](#) on page 229.

When getting an `Configuration` object for a HBase MapReduce job, instantiate it using the `HBaseConfiguration.create()` method.

## Troubleshooting

The Cloudera HBase packages have been configured to place logs in `/var/log/hbase`. Cloudera recommends tailing the `.log` files in this directory when you start HBase to check for any error messages or failures.

### Table Creation Fails after Installing LZO

If you install LZO after starting the Region Server, you will not be able to create a table with LZO compression until you re-start the Region Server.

#### Why this happens

When the Region Server starts, it runs `CompressionTest` and caches the results. When you try to create a table with a given form of compression, it refers to those results. You have installed LZO since starting the Region Server, so the cached results, which pre-date LZO, cause the create to fail.

#### What to do

Restart the Region Server. Now table creation with LZO will succeed.

## Viewing the HBase Documentation

For additional HBase documentation, see <http://archive.cloudera.com/cdh4/cdh/4/hbase/>.

## HBase Replication

HBase replication provides a means of copying the data from one HBase cluster to another (typically distant) HBase cluster. It is designed for data recovery rather than failover.

The cluster receiving the data from user applications is called the **master** cluster, and the cluster receiving the replicated data from the master is called the **slave** cluster.

### Types of Replication

You can implement any of the following replication models:

- [Master-slave replication](#)
- [Master-master replication](#)
- [Cyclic replication](#)

In all cases, the principle of replication is similar to that of MySQL master/slave replication in which each transaction on the master cluster is replayed on the slave cluster. In the case of HBase, the Write-Ahead Log (WAL) or HLog records all the transactions (Put/Delete) and the master cluster Region Servers ship the edits to the slave cluster Region Servers. This is done asynchronously, so having the slave cluster in a distant data center does not cause high latency at the master cluster.

#### Master-Slave Replication

This is the basic replication model, in which transactions on the master cluster are replayed on the slave cluster, as described above. For instructions on configuring master-slave replications, see [Deploying HBase Replication](#).

#### Master-Master Replication

In this case, the slave cluster in one relationship can act as the master in a second relationship, and the slave in the second relationship can act as master in a third relationship, and so on.

# HBase Installation

## Cyclic Replication

In the cyclic replication model, the slave cluster acts as master cluster for the original master. This sort of replication is useful when both the clusters are receiving data from different sources and you want each of these clusters to have the same data.

### ■ Important:

The normal configuration for cyclic replication is two clusters; you can configure more, but if you do, loop detection is not guaranteed in every case. Follow [these guidelines](#).

## Points to Note about Replication

- You make the configuration changes on the master cluster side.
- In the case of master-master replication, you make the changes on both sides.
- Replication works at the table-column-family level. The family should exist on all the slaves. (You can have additional, non replicating families on both sides).
- The timestamps of the replicated HLog entries are kept intact. In case of a collision (two entries identical as to row key, column family, column qualifier, and timestamp) only the entry arriving later will be read.
- Increment Column Values (ICVs) are treated as simple puts when they are replicated. In the master-master case, this may be undesirable, creating identical counters that overwrite one another. (See <https://issues.apache.org/jira/browse/HBase-2804>.)
- Make sure the master and slave clusters are time-synchronized with each other. Cloudera recommends you use Network Time Protocol (NTP).

## Requirements

Before configuring replication, make sure your environment meets the following requirements:

- You must manage Zookeeper yourself. It must not be managed by HBase, and must be available throughout the deployment.
- Each host in both clusters must be able to reach every other host, including those in the Zookeeper cluster.
- Both clusters must be running the same major version of CDH, for example CDH 4.x or CDH 5.x.
- Every table that contains families that are scoped for replication must exist on each cluster and have exactly the same name.
- HBase version 0.92 or greater is required for multiple slaves, master-master, or cyclic replication. This version ships with CDH4.0.0.

## Deploying HBase Replication

Follow these steps to enable replication from one cluster to another.

1. Edit `${HBASE_HOME}/conf/hbase-site.xml` on both clusters and add the following:

```
<property>
  <name>hbase.replication</name>
  <value>true</value>
</property>
```

2. Push `hbase-site.xml` to all nodes.
3. Restart HBase if it was running.
4. Run the following command in the HBase master's shell while it's running:

```
add_peer
```

This will show you the help for setting up the replication stream between the clusters. The command takes the form:

```
add_peer '<n>',
"slave.zookeeper.quorum:zookeeper.clientport.:zookeeper.znode.parent"
```

where <n> is the peer ID; it should not be more than two characters (longer IDs may work, but have not been tested).

Example:

```
hbase> add_peer '1', "zk.server.com:2181:/hbase"
```

■ **Note:**

If both clusters use the same Zookeeper cluster, you need to use a different `zookeeper.znode.parent` for each so that they don't write to the same directory.

5. Once you have a peer, enable replication on your column families. One way to do this is to alter the table and set the scope like this:

```
disable 'your_table'
alter 'your_table', {NAME => 'family_name', REPLICATION_SCOPE => '1'}
enable 'your_table'
```

Currently, a scope of 0 (default) means that the data will not be replicated and a scope of 1 means that it will. This could change in the future.

6. To list all configured peers, run the following command in the master's shell:

```
list_peers
```

You can confirm that your setup works by looking at any Region Server's log on the master cluster; look for the lines such as the following:

```
Considering 1 rs, with ratio 0.1
Getting 1 rs from peer cluster # 0
Choosing peer 170.22.64.15:62020
```

This indicates that one Region Server from the slave cluster was chosen for replication.

### Deploying Master-Master or Cyclic Replication

For master-master or cyclic replication, repeat the above steps on each master cluster: add the `hbase.replication` property and set it to `true`, push the resulting `hbase-site.xml` to all nodes of this master cluster, use `add_peer` to add the slave cluster, and enable replication on the column families.

### Guidelines for Replication across Three or More Clusters

When configuring replication among three or more clusters, follow these guidelines:

1. The replication relationships should have *either*:
  - **No loops** (as in A->B->C, or A->B->C->D, etc. ), *or*
  - If a loop exists, it should be a **complete cycle** (as in A->B->C->A, or A->B->C->D->A, etc.)
2. Cloudera recommends you enable `KEEP_DELETED_CELLS` on column families in the slave cluster, where `REPLICATION_SCOPE=1` in the master cluster; for example:

## HBase Installation

- On the master:

```
create 't1',{NAME=>'f1', REPLICATION_SCOPE=>1}
```

- On the slave:

```
create 't1',{NAME=>'f1', KEEP_DELETED_CELLS=>'true'}
```

### Disabling Replication at the Peer Level

Use the command `disable_peer (<"peerID">)` to disable replication for a specific peer. This will stop replication to the peer, but the logs will be kept for future reference.

To re-enable the peer, use the command `enable_peer (<"peerID">)`. Replication resumes where it was stopped.

#### Examples:

- To disable peer 1:

```
disable_peer("1")
```

- To re-enable peer 1:

```
enable_peer("1")
```

### Stopping Replication in an Emergency

If replication is causing serious problems, you can stop it while the clusters are running.

- **Warning:**

Do this only in case of a serious problem; it may cause data loss.

#### To stop replication in an emergency:

Open the shell on the master cluster and use the `stop_replication` command. For example:

```
hbase(main):001:0> stop_replication
```

Already queued edits will be replicated after you use the `stop_replication` command, but new entries will not.

To start replication again, use the `start_replication` command.

### Initiating Replication of Pre-existing Data

You may need to start replication from some point in the past. For example, suppose you have a primary HBase cluster in one location and are setting up a disaster-recovery (DR) cluster in another. To initialize the DR cluster, you need to copy over the existing data from the primary to the DR cluster, so that when you need to switch to the DR cluster you have a full copy of the data generated by the primary cluster. Once that is done, replication of new data can proceed as normal.

To start replication from an earlier point in time, run a `copyTable` command (defining the start and end timestamps), while enabling replication. Proceed as follows:

1. Start replication and note the timestamp.
2. Run the `copyTable` command with an end timestamp equal to the timestamp you noted in the previous step.



■ **Note:**

Because replication starts from the current WAL, some key values may be copied to the slave cluster by both the replication and the `copyTable` job. This is not a problem because this is an idempotent operation (one that can be applied multiple times without changing the result).

### Replicating Pre-existing Data in a Master-Master Deployment

In the case of master-master replication, run the `copyTable` job before starting the replication. (If you start the job after enabling replication, the second master will re-send the data to the first master, because `copyTable` does not edit the `clusterId` in the mutation objects. Proceed as follows:

1. Run the `copyTable` job and note the start timestamp of the job.
2. Start replication.
3. Run the `copyTable` job again with a start time equal to the start time you noted in step 1.

This results in some data being pushed back and forth between the two clusters; but it minimizes the amount of data.

### Verifying Replicated Data

If you are looking only at a few rows, you can verify the replicated data in the shell.

For a systematic comparison on a larger scale, use the `VerifyReplication` MapReduce job. Run it on the master cluster and provide it with the peer ID (the one you provided when establishing the replication stream), a table name, and a column family. Other options allow you to specify a time range and specific families. This job's short name is `verifyrep`; provide that name when pointing `hadoop jar` to the HBase JAR file.

The command has the following form:

```
hbase org.apache.hadoop.hbase.mapreduce.replication.VerifyReplication
[--starttime=timestamp] [--stoptime=timestamp] [--families=comma separated list of
families] <peerId> <tablename>
```

The command prints out `GOODROWS` and `BADROWS` counters; these correspond to replicated and non-replicated rows respectively.

### Configuring Secure HBase Replication

If you want to make HBase Replication secure, follow the instructions under [HBase Security Configuration](#).

### Caveats

- Two variables govern replication: `hbase.replication` as described above under [Deploying HBase Replication](#), and a replication `znode`. Stopping replication (using `stop_replication` as above) sets the `znode` to `false`. Two problems can result:
- If you add a new Region Server to the master cluster while replication is stopped, its current log will not be added to the replication queue, because the replication `znode` is still set to `false`. If you restart replication at this point (using `start_replication`), entries in the log will not be replicated.
- Similarly, if a log rolls on an existing Region Server on the master cluster while replication is stopped, the new log will not be replicated, because the the replication `znode` was set to `false` when the new log was created.
- Loop detection is not guaranteed in all cases if you use [cyclic replication](#) among more than two clusters. Follow [these guidelines](#)
- In the case of a long-running, write-intensive workload, the slave cluster may become unresponsive if its meta-handlers are blocked while performing the replication. CDH4.1 introduces three new properties to deal with this problem:

- `hbase.regionserver.replication.handler.count` - the number of replication handlers in the slave cluster (default is 3). Replication is now handled by separate handlers in the slave cluster to avoid the above-mentioned sluggishness. Increase it to a high value if the ratio of master to slave RegionServers is high.
- `replication.sink.client.retries.number` - the number of times the HBase replication client at the sink cluster should retry writing the WAL entries (default is 1).
- `replication.sink.client.ops.timeout` - the timeout for the HBase replication client at the sink cluster (default is 20 seconds).

## Configuring HBase Snapshots

### About HBase Snapshots

In previous HBase releases, the only way to a backup or to clone a table was to use `CopyTable` or `ExportTable`, or to copy all the `hfiles` in HDFS after disabling the table. The disadvantages of these methods are:

- `CopyTable` and `ExportTable` can degrade region server performance.
- Disabling the table means no reads or writes; this is usually unacceptable.

HBase Snapshots allow you to clone a table without making data copies, and with minimal impact on Region Servers. Exporting the table to another cluster should not have any impact on the the region servers.

### Use Cases

- Recovery from user or application errors
  - Useful because it may be some time before the database administrator notices the error

■ **Note:**

The database administrator needs to schedule the intervals at which to take and delete snapshots. Use a script or your preferred management tool for this; it is not built into HBase.

- The database administrator may want to save a snapshot right before a major application upgrade or change.

■ **Note:**

Snapshots are not primarily used for system upgrade protection because they would not roll back binaries, and would not necessarily be proof against bugs or errors in the system or the upgrade.

- Sub-cases for recovery:
  - Rollback to previous snapshot and merge in reverted data
  - View previous snapshots and selectively merge them into production

- Backup
  - Capture a copy of the database and store it outside HBase for disaster recovery
  - Capture previous versions of data for compliance, regulation, archiving
  - Export from snapshot on live system provides a more consistent view of HBase than `CopyTable` and `ExportTable`
- Audit and/or report view of data at a specific time
  - Capture monthly data for compliance

- Use for end-of-day/month/quarter reports
- Use for Application testing
  - Test schema or application changes on like production data from snapshot and then throw away
  - For example: take a snapshot; create a new table from the snapshot content (schema plus data); manipulate the new table by changing the schema, adding and removing rows, and so on (the original table, the snapshot, and the new table remain independent of each other)
- Offload work
  - Capture, copy, and restore data to another site
  - Export data to another cluster

### Where Snapshots Are Stored

The snapshot metadata is stored in the `.snapshot` directory under the `hbase` root directory (`/hbase/.snapshot`). Each snapshot has its own directory that includes all the references to the `hfiles`, logs, and metadata needed to restore the table.

`hfiles` needed by the snapshot are in the traditional `/hbase/<tableName>/<regionName>/<familyName>/` location if the table is still using them; otherwise they will be placed in `/hbase/.archive/<tableName>/<regionName>/<familyName>/`

### Zero-copy Restore and Clone Table

From a snapshot you can create a new table (`clone` operation) or restore the original table. These two operations do not involve data copies; instead a link is created to point to the original `hfiles`.

Changes to a cloned or restored table do not affect the snapshot or (in case of a clone) the original table.

If you want to clone a table to another cluster, you need to export the snapshot to the other cluster and then execute the `clone` operation; see [Exporting a Snapshot to Another Cluster](#).

### Reverting to a Previous HBase Version

Snapshots don't affect HBase backward compatibility if they are not used.

If you do use the snapshot capability, backward compatibility is affected as follows:

- If you only take snapshots, you can still go back to a previous HBase version
- If you have used `restore` or `clone`, you cannot go back to a previous version unless the cloned or restored tables have no links (there is no automated way to check; you would need to inspect the file system manually).

### Storage Considerations

Since the `hfiles` are immutable, a snapshot consists of reference to the files that are in the table at the moment the snapshot is taken. No copies of the data are made during the snapshot operation, but copies may be made when a compaction or deletion is triggered. In this case, if a snapshot has a reference to the files to be removed, the files are moved to an archive folder, instead of being deleted. This allows the snapshot to be restored in full.

Because no copies are performed, multiple snapshots share the same `hfiles`, but in the worst case scenario, each snapshot could have different set of `hfiles` (tables with lots of updates, and compactions).

## Configuring and Enabling Snapshots

Snapshots are off by default; to enable them, set the `hbase.snapshot.enabled` property in `hbase-site.xml` to `true`:

```
<property>
  <name>hbase.snapshot.enabled</name>
```

## HBase Installation

```
<value>
  true
</value>
</property>
```

To disable snapshots after you have enabled them, set `hbase.snapshot.enabled` to `false`.

■ **Note:**

If you have taken snapshots and then decide to disable snapshots, you must delete the snapshots before restarting the HBase master; the HBase master will not start if snapshots are disabled and snapshots exist.

Snapshots don't affect HBase performance if they are not used.

## Shell Commands

You can manage snapshots by using the HBase shell or the HBaseAdmin Java API.

The following table shows actions you can take from the shell:

Action	Shell command	Comments
Take a snapshot of <code>tableX</code> called <code>snapshotX</code>	<code>snapshot 'tableX', 'snapshotX'</code>	<p>Snapshots can be taken while a table is disabled, or while a table is online and serving traffic.</p> <ul style="list-style-type: none"> <li>▪ If a table is disabled (via <code>disable &lt;table&gt;</code>) an offline snapshot is taken. This snapshot is driven by the master and fully consistent with the state when the table was disabled. This is the simplest and safest method, but it involves a service interruption since the table must be disabled to take the snapshot.</li> <li>▪ In an online snapshot, the table remains available while the snapshot is taken, and should incur minimal noticeable performance degradation of normal read/write loads. This snapshot is coordinated by the master and run on the region servers. The current implementation - simple-flush snapshots - provides no causal consistency guarantees. Despite this shortcoming, it offers the same degree of consistency as <code>CopyTable</code> and overall is a huge improvement over <code>CopyTable</code>.</li> </ul>
Restore snapshot <code>snapshotX</code> (it will replace the source table content)	<code>restore_snapshot 'snapshotX'</code>	Restoring a snapshot attempts to replace the current version of a table with another version of the table. To run this command, you must disable the target table. The <code>restore</code> command takes a snapshot of the table (appending a timestamp code), and then essentially clones data into the original data and removes data not in the snapshot. If the operation succeeds, the target table will be enabled. Use this capability only in an emergency; see <a href="#">Restrictions</a> .
List all available snapshots	<code>list_snapshots</code>	
List all available snapshots starting with <code>'mysnapshot_'</code> (regular expression)	<code>list_snapshots 'my_snapshot_*'</code>	
Remove a snapshot called <code>snapshotX</code>	<code>delete_snapshot 'snapshotX'</code>	
Create a new table <code>tableY</code> from a		Cloning a snapshot creates a new

Action	Shell command	Comments
snapshot snapshotX	<code>clone_snapshot 'snapshotX', 'tableY'</code>	read/write table that can serve the data kept at the time of the snapshot. The original table and the cloned table can be modified independently without interfering – new data written to one table will not show up on the other.

## Exporting a Snapshot to Another Cluster

You can export any snapshot to another cluster. This involves the copy of the table `hfiles` and logs retained during the snapshot, and the snapshot metadata.

- **Note:** Snapshots must be enabled on the destination cluster. See [Configuring and Enabling Snapshots](#) on page 235.

The `ExportSnapshot` tool executes a MapReduce Job, similar to `distcp`, to copy files to the other cluster. It works at file-system level, so the hbase cluster can be offline.

**To copy a snapshot called `MySnapshot` to an HBase cluster `srv2` (`hdfs:///srv2:8082/hbase`) using 16 mappers:**

```
hbase class org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
hdfs:///srv2:8082/hbase -mappers 16
```

**To export the snapshot and change the ownership of the files during the copy:**

```
hbase class org.apache.hadoop.hbase.snapshot.ExportSnapshot -snapshot MySnapshot -copy-to
hdfs:///srv2:8082/hbase -chuser MyUser -chgroup MyGroup -chmod 700 -mappers 16
```

You can also use the Java `-D` option in many tools to specify MapReduce or other configuration. properties. For example, the following command copies `MY_SNAPSHOT` to `hdfs://cluster2/hbase` using groups of 10 `hfiles` per mapper:

```
hbase org.apache.hadoop.hbase.snapshot.ExportSnapshot
-Dsnapshot.export.default.map.group=10 -snapshot MY_SNAPSHOT -copy-to
hdfs://cluster2/hbase
```

(The number of mappers is calculated as `TotalNumberOfHFiles/10`.)

## Restrictions

- **Warning:**

**Do not use merge in combination with snapshots. Merging two regions can cause data loss if snapshots or cloned tables exist for this table.**

The merge is likely to corrupt the snapshot and any tables cloned from the snapshot. In addition, if the table has been restored from a snapshot, the merge may also corrupt the table. The snapshot may survive intact if the regions being merged are not in the snapshot, and clones may survive if they do not share files with the original table or snapshot. You can use the `Snapshot` tool (see [Information and Debugging](#) on page 239) to check the status of the snapshot. If the status is `BROKEN`, the snapshot is unusable.

- All the Masters and Region Servers must be running at least CDH4.2.
- If you have [enabled](#) the `AccessController` Coprocessor for HBase, only a global administrator can take, clone, or restore a snapshot, and these actions do not capture the ACL rights. This means that restoring a

table preserves the ACL rights of the existing table, while cloning a table creates a new table that has no ACL rights until the administrator adds them.

- Do not take, clone, or restore a snapshot during a rolling restart. Snapshots rely on the Region Servers being up; otherwise the snapshot will fail.

- **Note:** This restriction also applies to rolling upgrade, which can currently be done only via Cloudera Manager.

**If you are using HBase Replication and you need to restore a snapshot:** If you are using [HBase Replication](#) the replicas will be out of synch when you restore a snapshot. Do this only in an emergency.

- **Important:**

Snapshot restore is an emergency tool; you need to disable the table and [table replication](#) to get to an earlier state, and you may lose data in the process.

If you need to restore a snapshot, proceed as follows:

1. Disable the table that is the restore target, and stop the replication
2. Remove the table from both the master and slave clusters
3. Restore the snapshot on the master cluster
4. Create the table on the slave cluster and use `CopyTable` to initialize it.

- **Note:**

If this is not an emergency (for example, if you know that you have lost just a set of rows such as the rows starting with "xyz"), you can create a clone from the snapshot and create a MapReduce job to copy the data that you've lost.

In this case you don't need to stop replication or disable your main table.

## Snapshot Failures

Region moves, splits, and other metadata actions that happen while a snapshot is in progress will probably cause the snapshot to fail; the software detects and rejects corrupted snapshot attempts.

## Information and Debugging

You can use the `SnapshotInfo` tool to get information about a snapshot, including status, files, disk usage, and debugging information.

### Examples:

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot
Snapshot Info
-----
Name: test-snapshot
Type: DISABLED
Table: test-table
Version: 0
Created: 2012-12-30T11:21:21

*****
BAD SNAPSHOT: 1 hfile(s) and 0 log(s) missing.
*****
```

```
6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the source table)
0 Logs, total size 0.0
```

```
$ hbase org.apache.hadoop.hbase.snapshot.SnapshotInfo -snapshot test-snapshot -files
Snapshot Info
```

```
-----
Name: test-snapshot
Type: DISABLED
Table: test-table
Version: 0
Created: 2012-12-30T11:21:21
```

```
Snapshot Files
```

```
-----
52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/bdf29c39da2a4f2b81889eb4f7b18107
(archive)
52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/1e06029d0a2a4a709051b417aec88291
(archive)
86.8k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/506f601e14dc4c74a058be5843b99577
(archive)
52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/5c7f6916ab724eacbcea218a713941c4
(archive)
293.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/aec5e33a6564441d9bd423e31fc93abb
(archive)
52.4k test-table/02ba3a0f8964669520cf96bb4e314c60/cf/97782b2fbf0743edaacd8fef06ba51e4
(archive)
```

```
6 HFiles (6 in archive), total size 589.7k (0.00% 0.0 shared with the source table)
0 Logs, total size 0.0
```



## ZooKeeper Installation

- **Note: Running Services**

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

Apache ZooKeeper is a highly reliable and available service that provides coordination between distributed processes.

- **Note:**

**For More Information**

From the Apache ZooKeeper site:

"ZooKeeper is a high-performance coordination service for distributed applications. It exposes common services — such as naming, configuration management, synchronization, and group services - in a simple interface so you don't have to write them from scratch. You can use it off-the-shelf to implement consensus, group management, leader election, and presence protocols. And you can build on it for your own, specific needs."

To learn more about Apache ZooKeeper, visit <http://zookeeper.apache.org/>.

- **Note:**

To see which version of ZooKeeper is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

Use the following sections to install, upgrade and administer ZooKeeper:

- [Upgrading from CDH3](#)
- [Upgrading from an Earlier CDH4 Release](#)
- [Installing ZooKeeper](#)
- [Maintaining the Server](#)
- [Apache ZooKeeper Documentation](#)

## Upgrading ZooKeeper from CDH3 to CDH4

- **Important:**

**This affects you if you were running ZooKeeper under CDH3.** A change to the CDH4 ZooKeeper server RPM packaging has relocated the data directory default to `/var/lib/zookeeper` from the CDH3 default `/var/zookeeper`. Make sure you follow the instructions under [Step 4](#) below.

To upgrade ZooKeeper from CDH3 to CDH4, uninstall the CDH3 version (if you have not already done so) and then install the CDH4 version. Proceed as follows.

## ZooKeeper Installation

- **Note:** If you have already performed the steps to uninstall CDH3 described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with [Step 2](#).

### Step 1: Remove ZooKeeper

1. Stop the ZooKeeper server:

```
$ sudo service hadoop-zookeeper-server stop
```

or

```
$ sudo service hadoop-zookeeper stop
```

depending on the platform and release.

2. Remove CDH3 ZooKeeper

**To remove ZooKeeper on Red Hat-compatible systems:**

```
$ sudo yum remove hadoop-zookeeper-server  
$ sudo yum remove hadoop-zookeeper
```

**To remove ZooKeeper on Ubuntu and Debian systems:**

```
$ sudo apt-get purge hadoop-zookeeper-server  
$ sudo apt-get purge hadoop-zookeeper
```

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

**To remove ZooKeeper on SLES systems:**

```
$ sudo zypper remove hadoop-zookeeper-server  
$ sudo zypper remove hadoop-zookeeper
```

### Step 2: Install the ZooKeeper Base Package

See [Installing the ZooKeeper Base Package](#).

### Step 3: Install the ZooKeeper Server Package

See [Installing the ZooKeeper Server Package](#).

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

## Step 4: Edit `/etc/zookeeper/conf/zoo.cfg` or Move the Data Directory

### ■ Important:

After installing the new CDH4 version, you **must** complete this step.

Do one of the following:

- Edit the `/etc/zookeeper/conf/zoo.cfg` file and change the `dataDir` and `dataLogDir` properties to point to the directory that contains your CDH3 data; for example, change it from the CDH4 default `/var/lib/zookeeper` to the CDH3 default `/var/zookeeper` directory if that is where your CDH3 data resides. (Note that if the `dataLogDir` property is not explicitly specified, it defaults to the value of the `dataDir` property.) *or*
- Move the contents of the data directory (for example, `/var/zookeeper`) to `/var/lib/zookeeper`. Use a command such as:

```
sudo -u zookeeper cp -r /var/zookeeper/* /var/lib/zookeeper
```

In any case, after installing the new CDH4 ZooKeeper packages, verify that the `dataDir` (and potentially `dataLogDir`) specified in the CDH4 `/etc/zookeeper/conf/zoo.cfg` point to a valid ZooKeeper directory.

If you previously modified your CDH3 `zoo.cfg` configuration file (`/etc/zookeeper.dist/zoo.cfg`), RPM uninstall and re-install (using `yum remove` as in [Step 1](#)) renames and preserves a copy of your modified `zoo.cfg` as `/etc/zookeeper.dist/zoo.cfg.rpmsave`. You should compare this to the new `/etc/zookeeper/conf/zoo.cfg` and resolve any differences that should be carried forward (typically where you have changed property value defaults).

If your CDH3 `zoo.cfg` file has not been modified since installation, it will be auto-deleted when the CDH3 ZooKeeper package is removed.

## Step 5: Restart the Server

### ■ Important:

**Do not proceed** with restarting the server until you have completed [Step 4](#).

See [Installing the ZooKeeper Server Package](#) for instructions on starting the server.

## Upgrading ZooKeeper from an Earlier CDH4 Release

Cloudera recommends that you use a **rolling upgrade** process to upgrade ZooKeeper: that is, upgrade one server in the ZooKeeper ensemble at a time. This means bringing down each server in turn, upgrading the software, then restarting the server. The server will automatically rejoin the quorum, update its internal state with the current ZooKeeper leader, and begin serving client sessions.

This method allows you to upgrade ZooKeeper without any interruption in the service, and also lets you monitor the ensemble as the upgrade progresses, and roll back if necessary if you run into problems.

The instructions that follow assume that you are upgrading ZooKeeper as part of a CDH4 upgrade, and have already performed the steps under [Upgrading from an Earlier CDH4 Release](#).

### Performing a ZooKeeper Rolling Upgrade

Follow these steps to perform a rolling upgrade.

## ZooKeeper Installation

### Step 1: Stop the ZooKeeper Server on the First Node

To stop the ZooKeeper server:

```
$ sudo service zookeeper-server stop
```

### Step 2: Install the ZooKeeper Base Package on the First Node

See [Installing the ZooKeeper Base Package](#).

### Step 3: Install the ZooKeeper Server Package on the First Node

See [Installing the ZooKeeper Server Package](#).

■ **Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

### Step 4: Restart the Server

See [Installing the ZooKeeper Server Package](#) for instructions on starting the server.

The upgrade is now complete on this server and you can proceed to the next.

### Step 5: Upgrade the Remaining Nodes

Repeat Steps 1–4 above on each of the remaining nodes.

The ZooKeeper upgrade is now complete.

## Installing the ZooKeeper Packages

There are two ZooKeeper server packages:

- The `zookeeper` base package provides the basic libraries and scripts that are necessary to run ZooKeeper servers and clients. The documentation is also included in this package.
- The `zookeeper-server` package contains the `init.d` scripts necessary to run ZooKeeper as a daemon process. Because `zookeeper-server` depends on `zookeeper`, installing the server package automatically installs the base package.

■ **Note:**

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands to install ZooKeeper. For instructions, see [CDH4 Installation](#).

### Installing the ZooKeeper Base Package

To install ZooKeeper On Red Hat-compatible systems:

```
$ sudo yum install zookeeper
```

**To install ZooKeeper on Ubuntu and other Debian systems:**

```
$ sudo apt-get install zookeeper
```

**To install ZooKeeper on SLES systems:**

```
$ sudo zypper install zookeeper
```

## Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server

The instructions provided here deploy a single ZooKeeper server in "standalone" mode. This is appropriate for evaluation, testing and development purposes, but may not provide sufficient reliability for a production application. See [Installing ZooKeeper in a Production Environment](#) for more information.

**To install the ZooKeeper Server On Red Hat-compatible systems:**

```
$ sudo yum install zookeeper-server
```

**To install a ZooKeeper server on Ubuntu and other Debian systems:**

```
$ sudo apt-get install zookeeper-server
```

**To install ZooKeeper on SLES systems:**

```
$ sudo zypper install zookeeper-server
```

**To start ZooKeeper**

- **Note:**

ZooKeeper may start automatically on installation on Ubuntu and other Debian systems. This automatic start will happen only if the data directory exists; otherwise you will be prompted to initialize as shown below.

- **To start ZooKeeper after an upgrade from CDH3 or CDH4:**

- **Important:**

If you are upgrading from CDH3, **do not proceed** with restarting the server until you have completed [Step 4](#) of the upgrade procedure.

```
$ sudo service zookeeper-server start
```

- **To start ZooKeeper after a fresh install:**

```
$ sudo service zookeeper-server init
$ sudo service zookeeper-server start
```

- **Note:**

If you are deploying multiple ZooKeeper servers after a fresh install, you need to create a `myid` file in the data directory. You can do this by means of an `init` command option: `$ sudo service zookeeper-server init --myid=1`

## ZooKeeper Installation

### Installing ZooKeeper in a Production Environment

For use in a production environment, you should deploy ZooKeeper as an ensemble with an odd number of nodes. As long as a majority of the servers in the ensemble are available, the ZooKeeper service will be available. The minimum recommended ensemble size is three ZooKeeper servers, and it is recommended that each server run on a separate machine.

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 ensemble, and each server must also have a `myid` file in its data directory (by default `/var/lib/zookeeper`) that identifies it as one of the servers in the ensemble.

For instructions describing how to set up a multi-server deployment, see [Installing a Multi-Server Setup](#).

### Setting up Supervisory Process for the ZooKeeper Server

The ZooKeeper server is designed to be both highly reliable and highly available. This means that:

- If a ZooKeeper server encounters an error it cannot recover from, it will "fail fast" (the process will exit immediately)
- When the server shuts down, the ensemble remains active, and continues serving requests
- Once restarted, the server rejoins the ensemble without any further manual intervention.

Cloudera recommends that you fully automate this process by configuring a supervisory service to manage each server, and restart the ZooKeeper server process automatically if it fails. See the [ZooKeeper Administrator's Guide](#) for more information.

### Maintaining a ZooKeeper Server

The ZooKeeper server continually saves `znode` snapshot files and, optionally, transactional logs in a Data Directory to enable you to recover data. It's a good idea to back up the ZooKeeper Data Directory periodically. Although ZooKeeper is highly reliable because a persistent copy is replicated on each server, recovering from backups may be necessary if a catastrophic failure or user error occurs.

When you use the default configuration, the ZooKeeper server does not remove the snapshots and log files, so they will accumulate over time. You will need to clean up this directory occasionally, taking into account on your backup schedules and processes. To automate the cleanup, a `zkCleanup.sh` script is provided in the `bin` directory of the `zookeeper` base package. Modify this script as necessary for your situation. In general, you want to run this as a `cron` task based on your backup schedule.

The data directory is specified by the `dataDir` parameter in the ZooKeeper [configuration file](#), and the data log directory is specified by the `dataLogDir` parameter.

For more information, see [Ongoing Data Directory Cleanup](#).

### Viewing the ZooKeeper Documentation

For additional ZooKeeper documentation, see <http://archive.cloudera.com/cdh4/cdh/4/zookeeper/>.

## Whirr Installation

Apache Whirr is a set of libraries for running cloud services. You can use Whirr to run CDH4 clusters on cloud providers' clusters, such as Amazon Elastic Compute Cloud (Amazon EC2). There's no need to install the RPMs for CDH4 or do any configuration; a working cluster will start immediately with one command. It's ideal for running temporary Hadoop clusters to carry out a proof of concept, or to run a few one-time jobs. When you are finished, you can destroy the cluster and all of its data with one command.

- **Important:**

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands to install or update Whirr. For instructions, see [CDH4 Installation](#).

Use the following sections to install, upgrade, and deploy Whirr:

- [Upgrading Whirr](#)
- [Installing Whirr](#)
- [Generating an SSH Key Pair](#)
- [Defining a Cluster](#)
- [Launching a Cluster](#)
- [Apache Whirr Documentation](#)

## Upgrading Whirr

- **Note:**

To see which version of Whirr is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

### Upgrading Whirr from CDH3 to CDH4

To upgrade Whirr from CDH3, uninstall the CDH3 version, modify the properties file, and install the CDH4 version. Proceed as follows.

- **Note:**

If you have already performed the steps to uninstall CDH3 and all components, as described under [Upgrading from CDH3 to CDH4](#), you can skip Step 1 below and proceed with installing the new CDH4 version of Whirr.

#### Step 1: Remove Whirr

1. Stop the Whirr proxy. Kill the `hadoop-proxy.sh` process by pressing Control-C.
2. Destroy the Cluster. Whirr clusters are normally short-lived. If you have a running cluster, destroy it: see [Destroying a cluster](#).
3. Uninstall the CDH3 version of Whirr:

**On Red Hat-compatible systems:**

```
$ sudo yum remove whirr
```

### On SLES systems:

```
$ sudo zypper remove whirr
```

### On Ubuntu and Debian systems:

```
sudo apt-get purge whirr
```

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, **DO NOT PROCEED** before backing them up.

#### 4. Update the Properties File.

Edit the configuration file, called `hadoop.properties` in these instructions, and save it.

- For Hadoop, configure the following properties as shown:

- For MRv1:

```
whirr.env.repo=cdh4
whirr.hadoop.install-function=install_cdh_hadoop
whirr.hadoop.configure-function=configure_cdh_hadoop
```

- For YARN: see [Defining a Cluster](#).

- For HBase, configure the following properties as shown:

```
whirr.env.repo=cdh4
whirr.hadoop.install-function=install_cdh_hadoop
whirr.hadoop.configure-function=configure_cdh_hadoop
whirr.hbase.install-function=install_cdh_hbase
whirr.hbase.configure-function=configure_cdh_hbase
whirr.zookeeper.install-function=install_cdh_zookeeper
whirr.zookeeper.configure-function=configure_cdh_zookeeper
```

- For ZooKeeper, configure the following properties as shown:

```
whirr.env.repo=cdh4
whirr.zookeeper.install-function=install_cdh_zookeeper
whirr.zookeeper.configure-function=configure_cdh_zookeeper
```

See [Defining a Whirr Cluster](#) for a sample file.

- **Important:**

If you are upgrading from Whirr version 0.3.0, and are using an explicit image (AMI), make sure it comes from one of the supplied Whirr recipe files.

### Step 2: Install the new Version

See the next section, [Installing Whirr](#).

The upgrade is now complete. For more information, see [Defining a Whirr Cluster](#), [Launching a Cluster](#), and [Viewing the Whirr Documentation](#).



## Upgrading Whirr from an Earlier CDH4 Release to the Latest CDH4 Release

### Step 1: Stop the Whirr proxy.

Kill the `hadoop-proxy.sh` process by pressing Control-C.

### Step 2: Destroy the Cluster.

Whirr clusters are normally short-lived. If you have a running cluster, destroy it: see [Destroying a cluster](#).

### Step 3: Install the New Version of Whirr

See [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 SLES system:**

```
$ sudo zypper install whirr
```

**To install Whirr on another system:** Download a Whirr tarball from [here](#).

**To verify Whirr is properly installed:**

```
$ whirr version
```

## Generating an SSH Key Pair

After installing Whirr, generate a password-less SSH key pair to enable secure communication with the Whirr cluster.

```
ssh-keygen -t rsa -P ''
```

### ■ Note:

If you specify a non-standard location for the key files in the `ssh-keygen` command (that is, not `~/.ssh/id_rsa`), then you must specify the location of the private key file in the `whirr.private-key-file` property and the public key file in the `whirr.public-key-file` property. For more information, see the next section.

## Defining a Whirr Cluster

- Note:**

For information on finding your cloud credentials, see the [Whirr FAQ](#).

After generating an SSH key pair, the only task left to do before using Whirr is to define a cluster by creating a properties file. You can name the properties file whatever you like. The example properties file used in these instructions is named `hadoop.properties`. Save the properties file in your home directory. After defining a cluster in the properties file, you will be ready to launch a cluster and run MapReduce jobs.

- Important:**

The properties shown below are sufficient to get a bare-bones cluster up and running, but you will probably need to do more configuration to do real-life tasks, especially if you are using HBase and ZooKeeper. You can find more comprehensive template files in the `recipes` directory, for example `recipes/hbase-cdh.properties`.

### MRv1 Cluster

The following file defines a cluster with a single machine for the NameNode and JobTracker, and another machine for a DataNode and TaskTracker.

```
whirr.cluster-name=myhadoopcluster
whirr.instance-templates=1 hadoop-jobtracker+hadoop-namenode,1
hadoop-datanode+hadoop-tasktracker
whirr.provider=aws-ec2
whirr.identity=<cloud-provider-identity>
whirr.credential=<cloud-provider-credential>
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa.pub
whirr.env.repo=cdh4
whirr.hadoop-install-function=install_cdh_hadoop
whirr.hadoop-configure-function=configure_cdh_hadoop
whirr.hardware-id=m1.large
whirr.image-id=us-east-1/ami-ccb35ea5
whirr.location-id=us-east-1
```

### YARN Cluster

The following configuration provides the essentials for a YARN cluster. Change the number of instances for `hadoop-datanode+yarn-nodemanager` from 2 to a larger number if you need to.

```
whirr.cluster-name=myhadoopcluster
whirr.instance-templates=1 hadoop-namenode+yarn-resourcemanager+mapreduce-historyserver,2
hadoop-datanode+yarn-nodemanager
whirr.provider=aws-ec2
whirr.identity=<cloud-provider-identity>
whirr.credential=<cloud-provider-credential>
whirr.private-key-file=${sys:user.home}/.ssh/id_rsa
whirr.public-key-file=${sys:user.home}/.ssh/id_rsa.pub
whirr.env.mapreduce_version=2
whirr.env.repo=cdh4
whirr.hadoop.install-function=install_cdh_hadoop
whirr.hadoop.configure-function=configure_cdh_hadoop
whirr.mr_jobhistory.start-function=start_cdh_mr_jobhistory
whirr.yarn.configure-function=configure_cdh_yarn
whirr.yarn.start-function=start_cdh_yarn
whirr.hardware-id=m1.large
whirr.image-id=us-east-1/ami-ccb35ea5
whirr.location-id=us-east-1
```

## Launching a Cluster

**To launch a cluster:**

```
$ whirr launch-cluster --config hadoop.properties
```

As the cluster starts up, messages are displayed in the console. You can see debug-level log messages in a file named `whirr.log` in the directory where you ran the `whirr` command. After the cluster has started, a message appears in the console showing the URL you can use to access the web UI for Whirr.

## Running a Whirr Proxy

For security reasons, traffic from the network where your client is running is proxied through the master node of the cluster using an SSH tunnel (a SOCKS proxy on port 6666). A script to launch the proxy is created when you launch the cluster, and may be found in `~/.whirr/<cluster-name>`.

**To launch the Whirr proxy:**

1. Run the following command in a new terminal window:

```
$ . ~/.whirr/myhadoopcluster/hadoop-proxy.sh
```

2. To stop the proxy, kill the process by pressing Ctrl-C.

## Running a MapReduce job

After you launch a cluster, a `hadoop-site.xml` file is automatically created in the directory `~/.whirr/<cluster-name>`. You need to update the local Hadoop configuration to use this file.

**To update the local Hadoop configuration to use `hadoop-site.xml`:**

1. On all systems, type the following commands:

```
$ cp -r /etc/hadoop/conf.empty /etc/hadoop/conf.whirr
$ rm -f /etc/hadoop/conf.whirr/*-site.xml
$ cp ~/.whirr/myhadoopcluster/hadoop-site.xml /etc/hadoop/conf.whirr
```

2. If you are using an Ubuntu, Debian, or SLES system, type these commands:

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

3. If you are using a Red Hat system, type these commands:

```
$ sudo alternatives --install /etc/hadoop/conf hadoop-conf /etc/hadoop/conf.whirr
50
$ alternatives --display hadoop-conf
```

4. You can now browse HDFS:

```
$ hadoop fs -ls /
```

**To run a MapReduce job, run these commands:**

- For MRv1:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-0.20-mapreduce
$ hadoop fs -mkdir input
$ hadoop fs -put $HADOOP_MAPRED_HOME/CHANGES.txt input
```

## Whirr Installation

```
$ hadoop jar $HADOOP_MAPRED_HOME/hadoop-examples.jar wordcount input output
$ hadoop fs -cat output/part-* | head
```

- For YARN:

```
$ export HADOOP_MAPRED_HOME=/usr/lib/hadoop-mapreduce
$ hadoop fs -mkdir input
$ hadoop fs -put $HADOOP_MAPRED_HOME/CHANGES.txt input
$ hadoop jar $HADOOP_MAPRED_HOME/hadoop-mapreduce-examples.jar wordcount input output
$ hadoop fs -cat output/part-* | head
```

## Destroying a cluster

When you are finished using a cluster, you can terminate the instances and clean up the resources using the commands shown in this section.

### WARNING

All data will be deleted when you destroy the cluster.

#### To destroy a cluster:

1. Run the following command to destroy a cluster:

```
$ whirr destroy-cluster --config hadoop.properties
```

2. Shut down the SSH proxy to the cluster if you started one earlier.

## Viewing the Whirr Documentation

For additional documentation see the [Whirr Documentation](#).

## Snappy Installation

Snappy is a compression/decompression library. It aims for very high speeds and reasonable compression, rather than maximum compression or compatibility with other compression libraries. Use the following sections to install, upgrade, and use Snappy.

- [Upgrading Snappy](#)
- [Installing Snappy](#)
- [Using Snappy for MapReduce Compression](#)
- [Using Snappy for Pig Compression](#)
- [Using Snappy for Hive Compression](#)
- [Using Snappy Compression in Sqoop Imports](#)
- [Using Snappy Compression with HBase](#)
- [Apache Snappy Documentation](#)

## Upgrading Snappy to CDH4

To upgrade Snappy, simply install the `hadoop` package if you haven't already done so. This applies whether you are upgrading from CDH3 or from an earlier CDH4 release.

■ **Note:**

To see which version of Hadoop is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

## Snappy Installation

Snappy is provided in the `hadoop` package along with the other native libraries (such as native gzip compression).

To take advantage of Snappy compression you need to set certain configuration properties, which are explained in the following sections.

## Using Snappy for MapReduce Compression

It's very common to enable MapReduce intermediate compression, since this can make jobs run faster without you having to make any application changes. Only the temporary intermediate files created by Hadoop for the shuffle phase are compressed (the final output may or may not be compressed). Snappy is ideal in this case because it compresses and decompresses very fast compared to other compression algorithms, such as Gzip.

To enable Snappy for MapReduce intermediate compression for the whole cluster, set the following properties in `mapred-site.xml`:

- For MRv1:

```
<property>
  <name>mapred.compress.map.output</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compression.codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

## Snappy Installation

- For YARN:

```
<property>
  <name>mapreduce.map.output.compress</name>
  <value>true</value>
</property>
<property>
  <name>mapred.map.output.compress.codec</name>
  <value>org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

You can also set these properties on a per-job basis.

Use the properties in the following table to compress the final output of a MapReduce job. These are usually set on a per-job basis.

MRv1 Property	YARN Property	Description
mapred.output. compress	mapreduce.output. fileoutputformat.compress	Whether to compress the final job outputs (true or false)
mapred.output. compression.codec	mapreduce.output. fileoutputformat. compress.codec	If the final job outputs are to be compressed, which codec should be used. Set to <code>org.apache.hadoop.io.compress.SnappyCodec</code> for Snappy compression.
mapred.output. compression.type	mapreduce.output. fileoutputformat. compress.type	For <code>SequenceFile</code> outputs, what type of compression should be used (NONE, RECORD, or BLOCK). BLOCK is recommended.

- **Note:**

The MRv1 property names are also supported (though deprecated) in MRv2 (YARN), so it's not mandatory to update them in this release.

## Using Snappy for Pig Compression

Set the same properties for Pig as for MapReduce (see the table in the previous section).

## Using Snappy for Hive Compression

To enable Snappy compression for Hive output when creating `SequenceFile` outputs, use the following settings:

```
SET hive.exec.compress.output=true;
SET mapred.output.compression.codec=org.apache.hadoop.io.compress.SnappyCodec;
SET mapred.output.compression.type=BLOCK;
```

## Using Snappy compression in Sqoop Imports

- For Sqoop 1:

On the command line, use the following option to enable Snappy compression:

```
--compression-codec org.apache.hadoop.io.compress.SnappyCodec
```

It is a good idea to use the `--as-sequencefile` option with this compression option.

- **For Sqoop 2:**

When you create a job (`sqoop:000> create job`), choose 7 (SNAPPY) as the compression format.

## Using Snappy Compression with HBase

If you install Hadoop and HBase from RPM or Debian packages, Snappy requires no HBase configuration.

## Viewing the Snappy Documentation

For more information about Snappy, see <http://code.google.com/p/snappy/>.





## 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

▪ **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 [CDH4 Installation](#).

Use the following sections to install, update and use Mahout:

- [Upgrading Mahout](#)
- [Installing Mahout](#)
- [The Mahout Executable](#)
- [Getting Started](#)
- [The Apache Mahout Wiki](#)

## Upgrading Mahout

▪ **Note:**

To see which version of Mahout is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#).

### Upgrading Mahout from CDH3 to CDH4

To upgrade Mahout to CDH4, you must uninstall the CDH3 version and then install the CDH4 version. Proceed as follows.

#### Step 1: Remove CDH3 Mahout

**To remove Mahout on a Red Hat system:**

```
$ sudo yum remove mahout
```

**To remove Mahout on a SLES system:**

```
$ sudo zypper remove mahout
```

**To remove Mahout on an Ubuntu or Debian system:**

```
$ sudo apt-get purge mahout
```

- **Warning:**

If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to make sure the re-install succeeds, but be aware that `apt-get purge` removes all your configuration data. If you have modified any configuration files, DO NOT PROCEED before backing them up.

### Step 2: Install CDH4 Mahout

See [Installing Mahout](#).

- **Important:**

During uninstall, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`. During re-install, the package manager creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original CDH3 configuration file to the new CDH4 configuration file. In the case of Ubuntu and Debian upgrades, a file will not be installed if there is already a version of that file on the system, and you will be prompted to resolve conflicts; for details, see [Automatic handling of configuration files by dpkg](#).

### Upgrading Mahout from an Earlier CDH4 Release to the Latest CDH4 Release

To upgrade Mahout to the latest release, simply install the new version; see [Installing Mahout](#).

- **Important:**

During package upgrade, the package manager renames any configuration files you have modified from `<file>` to `<file>.rpmsave`, and creates a new `<file>` with applicable defaults. You are responsible for applying any changes captured in the original configuration file to the new configuration file. In the case of Ubuntu and Debian upgrades, you will be prompted if you have made changes to a file for which there is a new version; for details, see [Automatic handling of configuration files by dpkg](#).

## Installing Mahout

You can install Mahout from an RPM or Debian package, or from a tarball. 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 a SLES system:

```
$ sudo zypper install mahout
```

**To install Mahout on an Ubuntu or Debian system:**

```
$ sudo apt-get install mahout
```

**To access Mahout documentation:**

The Mahout docs are bundled in a `mahout-doc` package that should be installed separately. The contents of this package are saved under `/usr/share/doc/mahout*`.

```
$ sudo apt-get install mahout-doc
```

**To install Mahout on a system for which packages are not available:**

Download a Mahout tarball from [here](#).

## The Mahout Executable

The Mahout executable is installed in `/usr/bin/mahout`. Use this executable to run your analysis.

## Getting Started with Mahout

To get started with Mahout, you can follow the instructions in this [Apache Mahout Quickstart](#).

## The Mahout Documentation

For more information about Mahout, see the [Apache Mahout Wiki](#).



# HttpFS Installation

## ■ Note: Running Services

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

Use the following sections to install and configure HttpFS:

- [About HttpFS](#)
- [Packaging](#)
- [Prerequisites](#)
- [Installing HttpFS](#)
- [Configuring HttpFS](#)
- [Starting the Server](#)
- [Stopping the Server](#)
- [Using the Server with curl](#)

## About HttpFS

Apache Hadoop HttpFS is a service that provides HTTP access to HDFS.

HttpFS has a REST HTTP API supporting all HDFS File System operations (both read and write).

Common HttpFS use cases are:

- Read and write data in HDFS using HTTP utilities (such as `curl` or `wget`) and HTTP libraries from languages other than Java (such as Perl).
- Transfer data between HDFS clusters running different versions of Hadoop (overcoming RPC versioning issues), for example using Hadoop DistCp.
- Read and write data in HDFS in a cluster behind a firewall. (The HttpFS server acts as a gateway and is the only system that is allowed to send and receive data through the firewall).

HttpFS supports Hadoop pseudo-authentication, HTTP SPNEGO Kerberos, and additional authentication mechanisms via a plugin API. HttpFS also supports Hadoop proxy user functionality.

The `webhdfs` client file system implementation can access HttpFS via the Hadoop filesystem command (`hadoop fs`), by using Hadoop DistCp, and from Java applications using the Hadoop file system Java API.

The HttpFS HTTP REST API is interoperable with the WebHDFS REST HTTP API.

For more information about HttpFS, see

<http://archive.cloudera.com/cdh4/cdh/4/hadoop/hadoop-hdfs-httpfs/index.html>.

## HttpFS Packaging

There are two packaging options for installing HttpFS:

- The `hadoop-httpfs` RPM package

## HttpFS Installation

- The `hadoop-httpfs` Debian package

You can also download a Hadoop tarball, which includes HttpFS, from [here](#).

## HttpFS Prerequisites

Prerequisites for installing HttpFS are:

- A Unix-like system: see [CDH4 Requirements and Supported Versions](#) for details
- Java: see [Java Development Kit Installation](#) for details

- **Note:**

To see which version of HttpFS is shipping in CDH4, check the [Version and Packaging Information](#). For important information on new and changed components, see the [CDH4 Release Notes](#). CDH4 Hadoop works with the CDH4 version of HttpFS.

## Installing HttpFS

HttpFS is distributed in the `hadoop-httpfs` package. To install it, use your preferred package manager application. Install the package on the system that will run the HttpFS server.

- **Important:**

If you have not already done so, install Cloudera's Yum, zypper/YaST or Apt repository before using the following commands to install HttpFS. For instructions, see [CDH4 Installation](#).

**To install the HttpFS package on a Red Hat-compatible system:**

```
$ sudo yum install hadoop-httpfs
```

**To install the HttpFS server package on a SLES system:**

```
$ sudo zypper install hadoop-httpfs
```

**To install the HttpFS package on an Ubuntu or Debian system:**

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

- **Note:**

Installing the `httpfs` package creates an `httpfs` service configured to start HttpFS at system startup time.

You are now ready to configure HttpFS. See the [next section](#).

## Configuring HttpFS

When you install HttpFS from an RPM or Debian package, HttpFS creates all configuration, documentation, and runtime files in the standard Unix directories, as follows.

Type of File	Where Installed
Binaries	<code>/usr/lib/hadoop-httpfs/</code>
Configuration	<code>/etc/hadoop-httpfs/conf/</code>
Documentation	<i>for SLES:</i> <code>/usr/share/doc/packages/hadoop-httpfs/</code>
	<i>for other platforms:</i> <code>/usr/share/doc/hadoop-httpfs/</code>
Data	<code>/var/lib/hadoop-httpfs/</code>
Logs	<code>/var/log/hadoop-httpfs/</code>
temp	<code>/var/tmp/hadoop-httpfs/</code>
PID file	<code>/var/run/hadoop-httpfs/</code>

## Configuring the HDFS HttpFS Will Use

HttpFS reads the HDFS configuration from the `core-site.xml` and `hdfs-site.xml` files in `/etc/hadoop/conf/`. If necessary edit those files to configure the HDFS HttpFS will use.

## Configuring the HttpFS Proxy User

Edit `core-site.xml` and define the Linux user that will run the HttpFS server as a Hadoop proxy user. For example:

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

Then restart Hadoop to make the proxy user configuration active.

## Configuring HttpFS with Kerberos Security

To configure HttpFS with Kerberos Security, see [HttpFS Security Configuration](#).

## Starting the HttpFS Server

After you have completed all of the required configuration steps, you can start HttpFS:

```
$ sudo service hadoop-httpfs start
```

If you see the message `Server httpfs started!`, `status NORMAL` in the `httpfs.log` log file, the system has started successfully.

- **Note:**

By default, HttpFS server runs on port 14000 and its URL is `http://<HTTPFS_HOSTNAME>:14000/webhdfs/v1`.

## Stopping the HttpFS Server

To stop the HttpFS server:

```
$ sudo service hadoop-httpfs stop
```

## Using the HttpFS Server with curl

You can use a tool such as `curl` to access HDFS via HttpFS. For example, to obtain the home directory of the user `babu`, use a command such as this:

```
$ curl "http://localhost:14000/webhdfs/v1?op=gethomedirectory&user.name=babu"
```

You should see output such as this:

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie:
hadoop.auth="u=babu&p=babu&t=simple&e=1332977755010&s=JVfT4T785K4jeeLNWXK68rc/0xI=";
Version=1; Path=/
Content-Type: application/json
Transfer-Encoding: chunked
Date: Wed, 28 Mar 2012 13:35:55 GMT

{"Path": "\user\babu"}
```

See the [WebHDFS REST API web page](#) for complete documentation of the API.



## 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).

All components in CDH4 that produce or consume files support Avro data files as a file format. But bear in mind that because uniform Avro support is new, there may be some rough edges or missing features.

The following sections contain brief notes on how to get started using Avro in the various CDH4 components:

- [Avro Data Files](#)
- [Compression](#)
- [Flume](#)
- [Sqoop](#)
- [MapReduce](#)
- [Streaming](#)
- [Pig](#)
- [Hive](#)
- [Avro Tools](#)

## Avro Data Files

Avro data files have the `.avro` extension. Make sure the files you create have this extension, since some tools look for it to determine which files to process as Avro (e.g. `AvroInputFormat` and `AvroAsTextInputFormat` for MapReduce and Streaming).

## Compression

By default Avro data files are not compressed, but it is generally advisable to enable compression to reduce disk usage and increase read and write performance. Avro data files support Deflate and [Snappy](#) compression. Snappy is faster, while Deflate is slightly more compact.

You do not need to do any additional configuration to read a compressed Avro data file rather than an uncompressed one. However, to write an Avro data file you need to specify the type of compression to use. How you specify compression depends on the component being used, as explained in the sections below.

## Flume

The [HDFSEventSink](#) that is used to serialize event data onto HDFS supports plugin implementations of [EventSerializer](#) interface. Implementations of this interface have full control over the serialization format and can be used in cases where the default serialization format provided by the Sink does not suffice.

An abstract implementation of the EventSerializer interface is provided along with Flume, called the [AbstractAvroEventSerializer](#). This class can be extended to support custom schema for Avro serialization over

HDFS. A simple implementation that maps the events to a representation of String header map and byte payload in Avro is provided by the class [FlumeEventAvroEventSerializer](#) which can be used by setting the serializer property of the Sink as follows:

```
<agent-name>.sinks.<sink-name>.serializer = AVRO_EVENT
```

## Sqoop

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 snappy
```

■ **Note:**

Sqoop 2 does not currently support Avro.

## MapReduce

The Avro MapReduce API is an Avro module for running MapReduce programs which produce or consume Avro data files.

If you are using Maven, simply add the following dependency to your POM:

```
<dependency>
  <groupId>org.apache.avro</groupId>
  <artifactId>avro-mapred</artifactId>
  <version>1.7.3</version>
  <classifier>hadoop2</classifier>
</dependency>
```

Then write your program using the [Avro MapReduce javadoc](#) for guidance.

At runtime, include the `avro` and `avro-mapred` JARs in the `HADOOP_CLASSPATH`; and the `avro`, `avro-mapred` and `paranamer` JARs in `-libjars`.

To enable Snappy compression on output files call `AvroJob.setOutputCodec(job, "snappy")` when configuring the job. You will also need to include the `snappy-java` JAR in `-libjars`.

## Streaming

To read from Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroAsTextInputFormat` as the input format. This input format will convert each datum in the Avro data file to a string. For a "bytes" schema, this will be the raw bytes, while in the general case it will be a single-line [JSON](#) representation of the datum.

To write to Avro data files from a streaming program, specify `org.apache.avro.mapred.AvroTextOutputFormat` as the output format. This output format will create Avro data files with a "bytes" schema, where each datum is a tab-delimited key-value pair.

At runtime specify the `avro`, `avro-mapred` and `paranamer` JARs in `-libjars` in the streaming command.

To enable Snappy compression on output files, set the property `avro.output.codec` to `snappy`. You will also need to include the `snappy-java` JAR in `-libjars`.

## Pig

CDH provides `AvroStorage` for Avro integration in Pig.

To use it, first register the `piggybank` JAR file and supporting libraries:

```
REGISTER piggybank.jar
REGISTER lib/avro-1.7.3.jar
REGISTER lib/json-simple-1.1.jar
REGISTER lib/snappy-java-1.0.4.1.jar
```

Then you can load Avro data files as follows:

```
a = LOAD 'my_file.avro' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

Pig maps the Avro schema to a corresponding Pig schema.

You can store data in Avro data files with:

```
store b into 'output' USING org.apache.pig.piggybank.storage.avro.AvroStorage();
```

In the case of `store`, Pig generates an Avro schema from the Pig schema. It is possible to override the Avro schema, either by specifying it literally as a parameter to `AvroStorage`, or by using the same schema as an existing Avro data file. See the [Pig wiki](#) for details.

To store two relations in one script, specify an index to each `store` function. Here is an example:

```
set1 = load 'input1.txt' using PigStorage() as ( ... );
store set1 into 'set1' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
'1');

set2 = load 'input2.txt' using PigStorage() as ( ... );
store set2 into 'set2' using org.apache.pig.piggybank.storage.avro.AvroStorage('index',
'2');
```

For more information, see the [AvroStorage wiki](#); look for "index".

To enable Snappy compression on output files do the following before issuing the `STORE` statement:

```
SET mapred.output.compress true
SET mapred.output.compression.codec org.apache.hadoop.io.compress.SnappyCodec
SET avro.output.codec snappy
```

There is some additional documentation on the [Pig wiki](#). Note, however, that the version numbers of the JAR files to register are different on that page, so you should adjust them as shown above.

## Hive

The following example demonstrates how to create a Hive table that is backed by Avro data files:

```
CREATE TABLE doctors
ROW FORMAT
SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
STORED AS
INPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat'
TBLPROPERTIES ( 'avro.schema.literal'='{
  "namespace": "testing.hive.avro.serde",
```

```

    "name": "doctors",
    "type": "record",
    "fields": [
      {
        "name": "number",
        "type": "int",
        "doc": "Order of playing the role"
      },
      {
        "name": "first_name",
        "type": "string",
        "doc": "first name of actor playing role"
      },
      {
        "name": "last_name",
        "type": "string",
        "doc": "last name of actor playing role"
      },
      {
        "name": "extra_field",
        "type": "string",
        "doc": "an extra field not in the original file",
        "default": "fishfingers and custard"
      }
    ]
  }
}');

LOAD DATA LOCAL INPATH '/usr/share/doc/hive-0.7.1+42.55/examples/files/doctors.avro'
INTO TABLE doctors;

```

You could also create a Avro backed Hive table by using an Avro schema file:

```

CREATE TABLE my_avro_table(notused INT)
ROW FORMAT SERDE
'org.apache.hadoop.hive.serde2.avro.AvroSerDe'
WITH SERDEPROPERTIES (
  'avro.schema.url'='file:///tmp/schema.avsc')
STORED as INPUTFORMAT
'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT
'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat';

```

The `avro.schema.url` is a URL (here a `file://` URL) pointing to an Avro schema file that is used for reading and writing, it could also be an `hdfs://` URL, eg. `hdfs://hadoop-namenode-uri/examplefile`

To enable Snappy compression on output files, run the following before writing to the table:

```

SET hive.exec.compress.output=true;
SET avro.output.codec=snappy;

```

You will also need to include the `snappy-java` JAR in `--auxpath`. The `snappy-java` JAR is located at:

```

/usr/lib/hive/lib/snappy-java-1.0.4.1.jar

```

[Haivvreo SerDe](#) has been merged into Hive as `AvroSerDe`, and it is no longer supported in its original form. `schema.url` and `schema.literal` have been changed to `avro.schema.url` and `avro.schema.literal` as a result of the merge. If you were you using [Haivvreo SerDe](#), you can use the new Hive `AvroSerDe` with tables created with the `Haivvreo SerDe`. For example, if you have a table `my_avro_table` that uses the `Haivvreo SerDe`, you can do the following to make the table use the new `AvroSerDe`:

```

ALTER TABLE my_avro_table SET SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe';

ALTER TABLE my_avro_table SET FILEFORMAT
INPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerInputFormat'
OUTPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.AvroContainerOutputFormat';

```

## Avro Tools

Avro provides a set of tools for working with Avro data files and schemas. The tools are not (currently) packaged with CDH, but you can download the tools JAR from [Apache](#), and run it as follows to get a list of commands:

```
java -jar avro-tools-1.7.3.jar
```

See also [RecordBreaker](#) for information on turning text data into structured Avro data.



## Using the Parquet File Format with Impala, Hive, Pig, and MapReduce

Parquet is automatically installed when you install any of the above components, and the necessary libraries are automatically placed in the classpath for all of them. Copies of the libraries are in `/usr/lib/parquet` or inside the parcels in `/lib/parquet`.

The Parquet file format incorporates several features that make it highly suited to data warehouse-style operations:

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

Among components of the CDH distribution, Parquet support originated in Cloudera Impala. Impala can create Parquet tables, insert data into them, convert data from other file formats to Parquet, and then perform SQL queries on the resulting data files. Parquet tables created by Impala can be accessed by Hive, and vice versa.

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

## Using Parquet Tables with Impala

The Cloudera Impala component can create tables that use Parquet data files; insert data into those tables, converting the data into Parquet format; and query Parquet data files produced by Impala or by other components. The only syntax required is the `STORED AS PARQUETFILE` clause on the `CREATE TABLE` statement. After that, all `SELECT`, `INSERT`, and other statements recognize the Parquet format automatically. For example, a session in the `impala-shell` interpreter might look as follows:

```
[localhost:21000] > create table parquet_table (x int, y string) stored as parquetfile;
[localhost:21000] > insert into parquet_table select x, y from some_other_table;
Inserted 50000000 rows in 33.52s
[localhost:21000] > select y from parquet_table where x between 70 and 100;
```

Once you create a Parquet table this way in Impala, you can query it or insert into it through either Impala or Hive.

Remember that Parquet format is optimized for working with large data files, typically 1 GB each. Avoid using the `INSERT ... VALUES` syntax, or partitioning the table at too granular a level, if that would produce a large number of small files that cannot take advantage of the Parquet optimizations for large data chunks.

Inserting data into a partitioned Impala table can be a memory-intensive operation, because each data file requires a 1 GB memory buffer to hold the data before being written. Such inserts can also exceed HDFS limits on simultaneous open files, because each node could potentially write to a separate data file for each partition, all at the same time. Consider splitting up such insert operations into one `INSERT` statement per partition.

For complete instructions and examples, see [the Parquet section in the Impala documentation](#).

### Using Parquet Tables in Hive

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

```
hive> create table parquet_table_name (x INT, y STRING)
      ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
      STORED AS
        INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
        OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat";
```

■ **Note:**

Once you create a Parquet table this way in Hive, you can query it or insert into it through either Impala or Hive. Before the first time you access a newly created Hive table through Impala, issue a one-time `INVALIDATE METADATA` statement in the `impala-shell` interpreter to make Impala aware of the new table.

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

```
hive> create external table parquet_table_name (x INT, y STRING)
      ROW FORMAT SERDE 'parquet.hive.serde.ParquetHiveSerDe'
      STORED AS
        INPUTFORMAT "parquet.hive.DeprecatedParquetInputFormat"
        OUTPUTFORMAT "parquet.hive.DeprecatedParquetOutputFormat"
        LOCATION '/test-warehouse/tinytable';
```

To populate the table with an `INSERT` statement, and to read the table with a `SELECT` statement, see the [Impala documentation](#) for Parquet.

Select the compression to use when writing data with the `parquet.compression` property, for example:

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

The valid options for compression are:

- UNCOMPRESSED
- GZIP
- SNAPPY

### Using Parquet Files in Pig

#### Reading Parquet Files in Pig

Assuming the external table was created and populated with Impala or Hive as described above, the Pig instruction to read the data is:

```
grunt> A = LOAD '/test-warehouse/tinytable' USING parquet.pig.ParquetLoader AS (x: int, y int);
```

#### Writing Parquet Files in Pig

Create and populate a Parquet file with the `ParquetStorer` class:

```
grunt> store A into '/test-warehouse/tinytable' USING parquet.pig.ParquetStorer;
```



There are three compression options: `uncompressed`, `snappy`, and `gzip`. The default is `snappy`. You can specify one of them once before the first store instruction in a Pig script:

```
SET parquet.compression gzip;
```

■ **Note:**

With CDH 4.5 you must add thrift to Pig's additional JAR files:

```
export PIG_OPTS="-Dpig.additional.jars=$THRIFTJAR"
```

You can find thrift as follows:

```
if [ -e /opt/cloudera/parcels/CDH ] ; then
  CDH_BASE=/opt/cloudera/parcels/CDH
else
  CDH_BASE=/usr
fi
THRIFTJAR=`ls -l $CDH_BASE/lib/hive/lib/libthrift*.jar | awk '{print $9}' | head -1`
```

To use a Pig action involving Parquet files with Oozie, add thrift to the Oozie sharelib:

```
sudo -u oozie hdfs dfs -put $THRIFTJAR share/lib/pig
```

## Using Parquet Files in MapReduce

MapReduce needs thrift in its `CLASSPATH` and in `libjars` to access Parquet files. It also needs `parquet-format` in `libjars`. Perform the following setup before running MapReduce jobs that access Parquet data files:

```
if [ -e /opt/cloudera/parcels/CDH ] ; then
  CDH_BASE=/opt/cloudera/parcels/CDH
else
  CDH_BASE=/usr
fi
THRIFTJAR=`ls -l $CDH_BASE/lib/hive/lib/libthrift*.jar | awk '{print $9}' | head -1`
export HADOOP_CLASSPATH=$HADOOP_CLASSPATH:$THRIFTJAR
export LIBJARS=`echo "$CLASSPATH" | awk 'BEGIN { RS = ":" } { print }' | grep
parquet-format | tail -1`
export LIBJARS=$LIBJARS,$THRIFTJAR

hadoop jar my-parquet-mr.jar -libjars $LIBJARS
```

### Reading Parquet Files in MapReduce

Taking advantage of the `Example` helper classes in the Parquet JAR files, a simple map-only MapReduce job that reads Parquet files can use the `ExampleInputFormat` class and the `Group` value class. There is nothing special about the reduce phase when using Parquet files. The following example demonstrates how to read a Parquet file in a MapReduce job; portions of code specific to the Parquet aspect are shown in bold.

```
import static java.lang.Thread.sleep;
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.NullWritable;
```

```

import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.Mapper.Context;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.ExampleInputFormat;

public class TestReadParquet extends Configured
    implements Tool {
    private static final Log LOG =
        Log.getLog(TestReadParquet.class);

    /*
     * Read a Parquet record
     */
    public static class MyMap extends
        Mapper<LongWritable, Group, NullWritable, Text> {

        @Override
        public void map(LongWritable key, Group value, Context context) throws IOException,
            InterruptedException {
            NullWritable outKey = NullWritable.get();
            String outputRecord = "";
            // Get the schema and field values of the record
            String inputRecord = value.toString();
            // Process the value, create an output record
            // ...
            context.write(outKey, new Text(outputRecord));
        }
    }

    public int run(String[] args) throws Exception {

        Job job = new Job(getConf());

        job.setJarByClass(getClass());
        job.setJobName(getClass().getName());
        job.setMapOutputKeyClass(LongWritable.class);
        job.setMapOutputValueClass(Text.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(Text.class);
        job.setMapperClass(MyMap.class);
        job.setNumReduceTasks(0);

        job.setInputFormatClass(ExampleInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.setInputPaths(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        job.waitForCompletion(true);
        return 0;
    }

    public static void main(String[] args) throws Exception {
        try {
            int res = ToolRunner.run(new Configuration(), new TestReadParquet(), args);
            System.exit(res);
        } catch (Exception e) {
            e.printStackTrace();
            System.exit(255);
        }
    }
}

```

## Writing Parquet Files in MapReduce

When writing Parquet files you will need to provide a schema. The schema can be specified in the run method of the job before submitting it, for example:

```
...
import parquet.Log;
import parquet.example.data.Group;
import parquet.hadoop.example.GroupWriteSupport;
import parquet.hadoop.example.ExampleInputFormat;
import parquet.hadoop.example.ExampleOutputFormat;
import parquet.hadoop.metadata.CompressionCodecName;
import parquet.hadoop.ParquetFileReader;
import parquet.hadoop.metadata.ParquetMetadata;
import parquet.schema.MessageType;
import parquet.schema.MessageTypeParser;
import parquet.schema.Type;
...
public int run(String[] args) throws Exception {
    ...

    String writeSchema = "message example {\n" +
        "required int32 x;\n" +
        "required int32 y;\n" +
        "}";
    ExampleOutputFormat.setSchema(
        job,
        MessageTypeParser.parseMessageType(writeSchema));

    job.submit();
}
```

or it can be extracted from the input file(s) if they are in Parquet format:

```
import org.apache.hadoop.fs.FileSystem;
import org.apache.hadoop.fs.FileStatus;
import org.apache.hadoop.fs.LocatedFileStatus;
import org.apache.hadoop.fs.RemoteIterator;
...

public int run(String[]
    args) throws Exception {
    ...

    String inputFile = args[0];
    Path parquetFilePath = null;
    // Find a file in case a directory was passed

    RemoteIterator<LocatedFileStatus> it = FileSystem.get(getConf()).listFiles(new
    Path(inputFile), true);
    while(it.hasNext()) {
        FileStatus fs = it.next();

        if(fs.isFile()) {
            parquetFilePath = fs.getPath();
            break;
        }
    }
    if(parquetFilePath == null) {
        LOG.error("No file found for " + inputFile);
        return 1;
    }
    ParquetMetadata readFooter =
        ParquetFileReader.readFooter(getConf(), parquetFilePath);
    MessageType schema =
        readFooter.getFileMetaData().getSchema();
    GroupWriteSupport.setSchema(schema, getConf());

    job.submit();
}
```

## Using the Parquet File Format with Impala, Hive, Pig, and MapReduce

Records can then be written in the mapper by composing a Group as value using the Example classes and no key:

```
protected void map(LongWritable key, Text value,
    Mapper<LongWritable, Text, Void, Group>.Context context)
    throws java.io.IOException, InterruptedException {
    int x;
    int y;
    // Extract the desired output values from the input text
    //
    Group group = factory.newGroup()
        .append("x", x)
        .append("y", y);
    context.write(null, group);
}
```

Compression can be set before submitting the job with:

```
ExampleOutputFormat.setCompression(job, codec);
```

The codec should be one of the following:

- CompressionCodecName.UNCOMPRESSED
- CompressionCodecName.SNAPPY
- CompressionCodecName.GZIP

## Parquet File Interoperability

Impala has included Parquet support from the beginning, using its own high-performance code written in C++ to read and write the Parquet files. The Parquet JARs for use with Hive, Pig, and MapReduce are available with CDH 4.5 and higher. Using the Java-based Parquet implementation on a CDH release prior to CDH 4.5 is not supported.

A Parquet table created by Hive can typically be accessed by Impala 1.1.1 and higher with no changes, and vice versa. Prior to Impala 1.1.1, when Hive support for Parquet was not available, Impala wrote a dummy SerDes class name into each data file. These older Impala data files require a one-time `ALTER TABLE` statement to update the metadata for the SerDes class name before they can be used with Hive. See [the Impala Release Notes](#) for details.

A Parquet file written by Hive, Impala, Pig, or MapReduce can be read by any of the others. Different defaults for file and block sizes, compression and encoding settings, and so on might cause performance differences depending on which component writes or reads the data files. For example, Impala typically sets the HDFS block size to 1 GB and divides the data files into 1 GB chunks, so that each I/O request reads an entire data file.

There may be limitations in a particular release. The following are known limitations in CDH 4:

- The `TIMESTAMP` data type in Parquet files is not supported in Hive, Pig, or MapReduce in CDH 4. Attempting to read a Parquet table created with Impala that includes a `TIMESTAMP` column will fail.
- Parquet has not been tested with HCatalog. Without HCatalog, Pig cannot correctly read dynamically partitioned tables; that is true for all file formats.
- Currently, Impala does not support table columns using nested data types or composite data types such as maps, structs, or arrays. Any Parquet data files that include such types cannot be queried through Impala.

## Examples of Java Programs to Read and Write Parquet Files

You can find full examples of Java code at the Cloudera [Parquet examples](#) Github repository.

The [TestReadParquet.java](#) example demonstrates the "identity" transform. It reads any Parquet data file and writes a new file with exactly the same content.

The [TestReadWriteParquet.java](#) example reads a Parquet data file, and produces a new text file in CSV format with the same content.



# Sentry Installation

Sentry enables role-based, fine-grained authorization for HiveServer2 and Cloudera Impala. It provides classic database-style authorization for Hive and Impala. For more information, and instructions on configuring Sentry for Hive, see [Configuring Sentry](#).

## Installing Sentry

- **Important:**

If you have not already done so, install Cloudera's `yum`, `zypper`/`YaST` or `apt` repository before using the following commands. For instructions, see [CDH4 Installation](#).

1. Install Sentry as follows, depending on your operating system:

- On Red Hat and similar systems:

```
$ sudo yum install sentry
```

- On SLES systems:

```
$ sudo zypper install sentry
```

- On Ubuntu and Debian systems:

```
sudo apt-get update; sudo apt-get install sentry
```





## Maintenance Tasks and Notes

### Note: Running Services

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

This section provide instructions and information on the following:

- [Starting CDH Services](#)
- [Configuring init to Start Core Services](#)
- [Stopping CDH Services](#)
- [Uninstalling CDH Components](#)
- [SSH and HTTPS in a Hadoop Cluster](#)

## Starting CDH Services

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

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

START services in this order:

Order	Service	Comments	For instructions and more information
1	ZooKeeper	Cloudera recommends starting ZooKeeper before starting HDFS; this is a <b>requirement</b> in a <a href="#">high-availability (HA)</a> deployment. In any case, always start ZooKeeper before HBase.	<a href="#">Installing the ZooKeeper Server Package and Starting ZooKeeper on a Single Server</a> ; <a href="#">Installing ZooKeeper in a Production Environment</a> ; <a href="#">HDFS High Availability Initial Deployment</a> ; <a href="#">Configuring High Availability for the JobTracker (MRv1)</a>
2	HDFS	Start HDFS before all other services except ZooKeeper. If you are using HA, see the <a href="#">CDH4 High Availability Guide</a> for instructions.	<a href="#">Deploying HDFS on a Cluster</a> ; <a href="#">Configuring HDFS High Availability</a>
3	HttpFS		<a href="#">HttpFS Installation</a>
4a	MRv1	Start MapReduce before Hive or Oozie. Do not start MRv1 if YARN is running.	<a href="#">Deploying MapReduce v1 (MRv1) on a Cluster</a> ; <a href="#">Configuring High</a>

Order	Service	Comments	For instructions and more information
			<a href="#">Availability for the JobTracker (MRv1)</a>
4b	YARN	Start YARN before Hive or Oozie. Do not start YARN if MRv1 is running.	<a href="#">Deploying MapReduce v2 (YARN) on a Cluster</a>
5	HBase		<a href="#">Starting the HBase Master; Deploying HBase in a Distributed Cluster</a>
6	Hive	Start the Hive metastore before starting HiveServer2 and the Hive console.	<a href="#">Installing Hive</a> on page 192
7	Oozie		<a href="#">Starting the Oozie Server</a>
8	Flume 1.x		<a href="#">Running Flume</a>
9	Sqoop		<a href="#">Sqoop Installation</a> and <a href="#">Sqoop 2 Installation</a> on page 131
10	Hue		<a href="#">Hue Installation</a>

## Configuring init to Start Core Hadoop System Services

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

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

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

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

### ■ Important:

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

The `chkconfig` commands to use are:

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

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

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

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

- Important:**

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

The `chkconfig` commands to use are:

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

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

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

### Configuring init to Start Non-core Hadoop System Services

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

The `chkconfig` commands are:

Component	Server	Command
Hue	Hue server	<code>\$ sudo chkconfig hue on</code>
Oozie	Oozie server	<code>\$ sudo chkconfig oozie on</code>
HBase	HBase master	<code>\$ sudo chkconfig hbase-master on</code>
	On each HBase slave	<code>\$ sudo chkconfig hbase-regionserver on</code>
Hive metastore HiveServer2	Hive server	<code>\$ sudo chkconfig hive-metastore on</code> <code>\$ sudo chkconfig hive-server2 on</code>
Zookeeper	Zookeeper server	<code>\$ sudo chkconfig zookeeper-server on</code>
HttpFS	HttpFS server	<code>\$ sudo chkconfig hadoop-httpfs on</code>

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

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

## Stopping Services

Run the following command on every host in the cluster to shut down all Hadoop Common system services that are started by `init` in the cluster:

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

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

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

You could also use `ps -fu hdfs` and `ps -fu mapred` to confirm that no processes are running as the `hdfs` or `mapred` user, but additional user names are created by the other components in the ecosystem; checking for the "java" string in the `ps` output is an easy way to identify any processes that may still be running.

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

STOP system services in this order:

Order	Service	Comments	Instructions
1	Hue		Run the following on the Hue Server machine to stop Hue <code>sudo service hue stop</code>
2	Sqoop		Run the following on all nodes where it is running:

Order	Service	Comments	Instructions
			<code>sudo service sqoop-metastore stop</code>
3	Flume 0.9		<p>Stop the Flume Node processes on each node where they are running:</p> <pre>sudo service flume-node stop</pre> <p>Stop the Flume Master</p> <pre>sudo service flume-master stop</pre>
4	Flume 1.x	There is no Flume master	<p>Stop the Flume Node processes on each node where they are running:</p> <pre>sudo service flume-ng-agent stop</pre>
5	Oozie		<code>sudo service oozie stop</code>
6	Hive		<p>To stop Hive, exit the Hive console and make sure no Hive scripts are running.</p> <p>Shut down HiveServer2:</p> <pre>sudo service hiveserver2 stop</pre> <p>Shut down the Hive metastore daemon on each client:</p> <pre>sudo service hive-metastore stop</pre> <p>If the metastore is running from the command line, use Ctrl-c to shut it down.</p>
7	HBase	Stop the Thrift server and clients, then shut down the cluster.	<p>To stop the Thrift server and clients:</p> <pre>sudo service hbase-thrift stop</pre> <p>To shut down the cluster, use this command on the master node:</p> <pre>sudo service hbase-master stop</pre> <p>Use the following command on each node hosting a region server:</p> <pre>sudo service hadoop-hbase-regionserver stop</pre>

Order	Service	Comments	Instructions
8a	MapReduce v1	Stop Hive and Oozie before stopping MapReduce.	To stop MapReduce, stop the JobTracker service, and stop the Task Tracker on all nodes where it is running. Use the following commands:  <pre>sudo service <del>hadoop-0.20-mapreduce-jobtracker</del> stop</pre> <pre>sudo service <del>hadoop-0.20-mapreduce-tasktracker</del> stop</pre>
8b	YARN	Stop Hive and Oozie before stopping YARN.	To stop YARN, stop the MapReduce JobHistory service, ResourceManager service, and NodeManager on all nodes where they are running. Use the following commands:  <pre>sudo service <del>hadoop-mapreduce-historyserver</del> stop</pre> <pre>sudo service hadoop-yarn-resourcemanager stop sudo service hadoop-yarn-nodemanager stop</pre>
9	HttpFS		<div style="border: 1px dashed black; padding: 5px;"> <pre>sudo service hadoop-httpfs stop</pre> </div>
10	HDFS		To stop HDFS: On the NameNode: <pre>sudo service hadoop-hdfs-namenode stop</pre> On the Secondary NameNode (if used): <pre>sudo service <del>hadoop-hdfs-secondarynamenode</del> stop</pre> On each DataNode: <pre>sudo service hadoop-hdfs-datanode stop</pre>
11	ZooKeeper	Stop HBase and HDFS before stopping ZooKeeper.	To stop the ZooKeeper server, use one of the following commands on each ZooKeeper node:



Order	Service	Comments	Instructions
			<pre>sudo service zookeeper-server stop</pre> <p>or</p> <pre>sudo service zookeeper stop</pre>

## Uninstalling CDH Components

Before uninstalling CDH, stop all Hadoop processes, following the instructions in [Stopping Services](#).

Here are the commands to use to uninstall the Hadoop components on different Linux systems.

Operating System	Commands	Comments
Red-Hat-compatible	<code>yum remove</code>	
Debian and Ubuntu	<code>apt-get purge</code> or <code>apt-get remove</code>	<code>apt-get</code> can be run with the <code>remove</code> option to remove only the installed packages or with the <code>purge</code> option to remove packages and configuration
SLES	<code>zypper remove</code>	

## Uninstalling from Red Hat, CentOS, and Similar Systems

Component to remove	Command
Mahout	<code>\$ sudo yum remove mahout</code>
Whirr	<code>\$ sudo yum remove whirr</code>
Hue	<code>\$ sudo yum remove hue</code>
Pig	<code>\$ sudo yum remove pig</code>
Sqoop	<code>\$ sudo yum remove sqoop</code>
Flume	<code>\$ sudo yum remove flume</code>
Oozie client	<code>\$ sudo yum remove oozie-client</code>
Oozie server	<code>\$ sudo yum remove oozie</code>
Hive	<code>\$ sudo yum remove hive hive-metastore hive-server hive-server2</code>
HBase	<code>\$ sudo yum remove hadoop-hbase</code>
ZooKeeper server	<code>\$ sudo yum remove hadoop-zookeeper-server</code>
ZooKeeper client	<code>\$ sudo yum remove hadoop-zookeeper</code>
ZooKeeper Failover Controller (ZKFC)	<code>\$ sudo yum remove hadoop-hdfs-zkfc</code>

Component to remove	Command
HDFS HA Journal Node	<code>\$ sudo yum remove hadoop-hdfs-hadoop-hdfs-journalnode</code>
Hadoop repository packages	<code>\$ sudo yum remove cloudera-cdh3</code>
HttpFS	<code>\$ sudo yum remove hadoop-httpfs</code>
Hadoop core packages	<code>\$ sudo yum remove hadoop-0.20</code>

## Uninstalling from Debian and Ubuntu

Use the `apt-get` command to uninstall software on Debian and Ubuntu systems. You can use `apt-get remove` or `apt-get purge`; the difference is that `apt-get purge` removes all your configuration data as well as the package files. If you are upgrading an Ubuntu or Debian system from CDH3u3 or earlier, you **must** use `apt-get purge` (rather than `apt-get remove`) to ensure that the re-install succeeds.

The `purge` commands to uninstall the Hadoop components from a Debian or Ubuntu system are:

Component to remove	Command
Mahout	<code>\$ sudo apt-get purge mahout</code>
Whirr	<code>\$ sudo apt-get purge whirr</code>
Hue	<code>\$ sudo apt-get purge hue</code>
Pig	<code>\$ sudo apt-get purge pig</code>
Sqoop	<code>\$ sudo apt-get purge sqoop</code>
Flume	<code>\$ sudo apt-get remove flume</code>
Oozie client	<code>\$ sudo apt-get purge oozie-client</code>
Oozie server	<code>\$ sudo apt-get purge oozie</code>
Hive	<code>\$ sudo apt-get purge hive hive-metastore hive-server hive-server2</code>
HBase	<code>\$ sudo apt-get purge hadoop-hbase</code>
ZooKeeper server	<code>\$ sudo apt-get purge hadoop-zookeeper-server</code>
ZooKeeper client	<code>\$ sudo apt-get purge hadoop-zookeeper</code>
ZooKeeper Failover Controller (ZKFC)	<code>\$ sudo apt-get purge hadoop-hdfs-zkfc</code>
HDFS HA Journal Node	<code>\$ apt-get purge hadoop-hdfs-hadoop-hdfs-journalnode</code>
HttpFS	<code>\$ sudo apt-get purge hadoop-httpfs</code>
Hadoop repository packages	<code>\$ sudo apt-get remove cdh3-repository</code>
Hadoop core packages	<code>\$ sudo apt-get purge hadoop-0.20</code>

## Uninstalling from SLES

Component removed	Command
Mahout	<code>\$ sudo zypper remove mahout</code>
Whirr	<code>\$ sudo zypper remove whirr</code>
Hue	<code>\$ sudo zypper remove hue</code>
Pig	<code>\$ sudo zypper remove pig</code>
Sqoop	<code>\$ sudo zypper remove sqoop</code>
Flume	<code>\$ sudo zypper remove flume</code>
Oozie server	<code>\$ sudo zypper remove oozie</code>
Oozie client	<code>\$ sudo zypper remove oozie-client</code>
Hive	<code>\$ sudo zypper remove hive hive-metastore hive-server hive-server2</code>
HBase	<code>\$ sudo zypper remove hadoop-hbase</code>
ZooKeeper server	<code>\$ sudo zypper remove hadoop-zookeeper-server</code>
ZooKeeper client	<code>\$ sudo zypper remove hadoop-zookeeper</code>
ZooKeeper Failover Controller (ZKFC)	<code>\$ sudo zypper remove hadoop-hdfs-zkfc</code>
HDFS HA Journal Node	<code>\$ sudo zypper remove hadoop-hdfs-hadoop-hdfs-journalnode</code>
HttpFS	<code>\$ sudo zypper remove hadoop-httpfs</code>
Hadoop repository packages	<code>\$ sudo zypper remove cloudera-cdh</code>
Hadoop core packages	<code>\$ sudo zypper remove hadoop-0.20</code>

## Additional clean-up

The uninstall commands may not remove all traces of Hadoop from your system. The `apt-get purge` commands available for Debian and Ubuntu systems delete more files than the commands that use the `remove` option but are still not comprehensive. If you want to remove all vestiges of Hadoop from your system, look for the following and remove them manually:

- log files
- modified system configuration files
- Hadoop configuration files in directories under `/etc` such as `hadoop`, `hbase`, `hue`, `hive`, `oozie`, `sqoop`, `zookeeper`, and `zookeeper.dist`
- user/group identifiers
- Oozie and Hue databases
- Documentation packages

## SSH and HTTPS in the Hadoop Cluster

SSH and HTTPS can be used to transmit information securely:

## Maintenance Tasks and Notes

- **SSH (Secure Shell)** is a secure shell that usually runs on top of SSL and has a built-in username/password authentication scheme that can be used for secure access to a remote host; it is a more secure alternative to `rlogin` and `telnet`.
- **HTTPS (HTTP Secure)** is HTTP running on top of SSL, adding security to standard HTTP communications.

### SSH

It is a good idea to use SSH for remote administration purposes (instead of `rlogin`, for example). But note that it is not used to secure communication among the elements in a Hadoop cluster (DataNode, NameNode, TaskTracker or YARN ResourceManager, JobTracker or YARN NodeManager, or the `/etc/init.d` scripts that start daemons locally).

The Hadoop components use SSH in the following cases:

- The [sshfencer](#) component of High Availability Hadoop configurations uses SSH; the `shell` fencing method does not require SSH.
- Whirr uses SSH to enable secure communication with the Whirr cluster in the Cloud. See the [Whirr Installation instructions](#).

### HTTPS

Some communication within Hadoop can be configured to use HTTPS. Implementing this requires generating valid certificates and configuring clients to use those certificates. The HTTPS functionality that can be configured in CDH4 is:

- Encrypted MapReduce Shuffle (both MRv1 and YARN).
- Encrypted Web UIs; the same configuration parameters that enable Encrypted MapReduce Shuffle implement Encrypted Web UIs.

These features are discussed under [Configuring Encrypted Shuffle, Encrypted Web UIs, and Encrypted HDFS Transport](#).

## The HDFS Balancer

The HDFS balancer re-balances data across the DataNodes, moving blocks from over-utilized to under-utilized nodes. As the system administrator, you can run the balancer from the command-line as necessary -- for example, after adding new DataNodes to the cluster.

Points to note:

- The balancer requires the capabilities of an HDFS superuser (for example, the `hdfs` user) to run.
- You can run the balancer without parameters, as follows:

```
sudo -u hdfs hdfs balancer
```

#### ■ Note:

If [Kerberos is enabled](#), do not use commands in the form `sudo -u <user> hadoop <command>`; they will fail with a security error. Instead, use the following commands: `$ kinit <user>` (if you are using a password) or `$ kinit -kt <keytab> <principal>` (if you are using a keytab) and then, for each command executed by this user, `$ <command>`

This runs the balancer with a default threshold of 10%, meaning that the script will ensure that disk usage on each DataNode differs from the overall usage in the cluster by no more than 10%. For example, if overall usage across all the DataNodes in the cluster is 40% of the cluster's total disk-storage capacity, the script ensures that each DataNode's disk usage is between 30% and 50% of that DataNode's disk-storage capacity.

- You can run the script with a different threshold; for example:

```
sudo -u hdfs hdfs balancer -threshold 5
```

This specifies that each DataNode's disk usage must be (or will be adjusted to be) within 5% of the cluster's overall usage.

- You can adjust the network bandwidth used by the balancer, by running the `dfsadmin -setBalancerBandwidth` command before you run the balancer; for example:

```
dfsadmin -setBalancerBandwidth newbandwidth
```

where *newbandwidth* is the maximum amount of network bandwidth, in bytes per second, that each DataNode can use during the balancing operation. For more information about the bandwidth command, see [this page](#).

- The balancer can take a long time to run, especially if you are running it for the first time, or do not run it regularly.

For more information, see the [Balancer Administrator Guide](#).



## Mountable HDFS

CDH4 includes a FUSE (Filesystem in Userspace) interface into HDFS. FUSE enables you to write a normal userland application as a bridge for a traditional filesystem interface. The `hadoop-hdfs-fuse` package enables you to use your HDFS cluster as if it were a traditional filesystem on Linux. It is assumed that you have a working HDFS cluster and know the hostname and port that your NameNode exposes.

**To install fuse-dfs On Red Hat-compatible systems:**

```
$ sudo yum install hadoop-hdfs-fuse
```

**To install fuse-dfs on Ubuntu systems:**

```
$ sudo apt-get install hadoop-hdfs-fuse
```

**To install fuse-dfs on SLES systems:**

```
$ sudo zypper install hadoop-hdfs-fuse
```

You now have everything you need to begin mounting HDFS on Linux. **To set up and test your mount point:**

```
$ mkdir -p <mount_point>
$ hadoop-fuse-dfs dfs://<name_node_hostname>:<namenode_port> <mount_point>
```

You can now run operations as if they are on your mount point. Press Ctrl+C to end the `hadoop-fuse-dfs` program, and `umount` the partition if it is still mounted.

■ **Note:**

If you are using SLES 11 with the Oracle JDK 6u26 package, `hadoop-fuse-dfs` may exit immediately because `ld.so` can't find `libjvm.so`. To work around this issue, add `/usr/java/latest/jre/lib/amd64/server` to the `LD_LIBRARY_PATH`.

**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.

## Mountable HDFS

You can change the JVM maximum heap size. By default, the CDH4 package installation creates the `/etc/default/hadoop-fuse` file with the following default JVM maximum heap size. To change it:

```
export LIBHDFS_OPTS="-Xmx128m"
```

For more information, see the help for `hadoop-fuse-dfs`:

```
$ hadoop-fuse-dfs --help
```



## Java Development Kit Installation

Install the Oracle Java Development Kit (JDK) before deploying CDH4.

- To install the JDK, follow the instructions under [Oracle JDK Installation](#). The completed installation must meet the requirements in the box below.
- If you have already installed a version of the JDK, make sure your installation meets the requirements in the box below.

■ **Note: Requirements**

- See [CDH4 Requirements and Supported Versions](#) for the recommended and supported JDK versions.
- If you are deploying CDH on a cluster, the same version of the Oracle JDK must be installed on each node.
- The JDK should be installed in `/usr/java/jdk-version`.
  - In addition, to make sure the right version of the JDK is always used, follow [these directions](#) these directions.

After [installing the JDK](#), and **before installing and deploying CDH**:

You may be able to install the Oracle JDK with your package manager, depending on your choice of operating system.

See [this section](#) for installation instructions.

## Oracle JDK Installation

■ **Important:**

The Oracle JDK installer is available both as an RPM-based installer (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, or `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 Oracle JDK:**

1. Download the `.tar.gz` file for one of the [recommended versions](#) of the Oracle JDK from [the Oracle Java SE page](#).
2. Extract the JDK to `/usr/java/jdk-version`; for example `/usr/java/1.7.0_15`
3. In `/etc/default/bigtop-utils`, set `JAVA_HOME` to the directory where the JDK is installed; for example:

```
export JAVA_HOME=/usr/java/default
```

4. Symbolically link the directory where the JDK is installed to `/usr/java/default`; for example:

```
ln -s /usr/java/jdk.1.7.0_15 /usr/java/default
```



## Creating a Local Yum Repository

This section explains how to set up a local `yum` repository which you can then use to install CDH on the machines in your cluster. There are a number of reasons you might want to do this, for example:

- The computers in your cluster may not have Internet access. You can still use `yum` to do an installation on those machines by creating a local `yum` repository.
- You may want to keep a stable local repository to ensure that any new installations (or re-installations on existing cluster members) use exactly the same bits.
- Using a local repository may be the most efficient way to distribute the software to the cluster members.

To set up your own internal mirror, do the following.

■ **Note:**

**Before You Start**

These instructions assume you already have the appropriate Cloudera repo file on the system on which you are going to create the local repository. If this is not the case, follow the instructions under [To download and install the CDH4 Package](#). (Downloading and installing the RPM also downloads the repo file and saves it in `/etc/yum.repos.d/`.)

1. On a computer that *does* have Internet access, install a web server such as `apache/lighttpd` on the machine which will serve the RPMs. The default configuration should work. Make sure the firewall on this web server will let http traffic go through.
2. On the same computer as in the previous step, install the `yum-utils` and `createrepo` packages if they are not already installed (`yum-utils` includes the `reposync` command):

```
sudo yum install yum-utils createrepo
```

3. On the same computer as in the previous steps, download the `yum` repository into a temporary location. On Red Hat/CentOS 6, you can use a command such as:

```
reposync -r cloudera-cdh4
```

■ **Note:**

`cloudera-cdh4` is the name of the repository on your system; the name is usually in square brackets on the first line of the repo file, which in this example is `/etc/yum.repos.d/cloudera-cdh4.repo`.

4. Put all the RPMs into a directory served by your web server. For this example, we'll call it `/var/www/html/cdh/4/RPMS/noarch/` (or `x86_64` or `i386` instead of `noarch`). Make sure you can remotely access the files in the directory you just created (the URL should look like `http://<yourwebserver>/cdh/4/RPMS/`).
5. On your web server, go to `/var/www/html/cdh/4/` and type the following command:

```
createrepo .
```

This will create or update the necessary metadata so `yum` can understand this new repository (you will see a new directory named `repodata`).

■ **Important:**

Check the permissions of the subdirectories and files under `/var/www/html/cdh/4/`. Make sure they are all readable by your web server user.

6. Edit the repo file you got from Cloudera (see [Before You Start](#)) and replace the line starting with `baseurl=` or `mirrorlist=` with `baseurl=http://<yourwebserver>/cdh/4/`
7. Save this modified repo file in `/etc/yum.repos.d/`, and check that you can install CDH through `yum`.

**Example:**

```
yum update && yum install hadoop
```

Once you have confirmed that your internal mirror works, you can distribute this modified repo file to all your machines, and they should all be able to install CDH without needing access to the Internet. Follow the instructions under [CDH4 Installation](#).

## Using the CDH4 Maven Repository

If you want to build applications or tools with the CDH4 components and you are using Maven or Ivy for dependency management, you can pull the CDH4 artifacts from the Cloudera Maven repository. The repository is available at <https://repository.cloudera.com/artifactory/cloudera-repos/>. The following is a sample POM (pom.xml) file:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">

  <repositories>
    <repository>
      <id>cloudera</id>
      <url>https://repository.cloudera.com/artifactory/cloudera-repos/</url>
    </repository>
  </repositories>

</project>
```

For detailed information for each CDH component, by release, see [Using the CDH4 Maven Repository](#).



## Building RPMs from CDH Source RPMs

This section describes how to build binary packages (RPMs) from published CDH source packages (SRPMs):

- [Prerequisites](#)
- [Setting up an Environment for Building RPMs](#)
- [Building an RPM](#)

### Prerequisites

- Oracle Java Development Kit (JDK) version 6.
- [Apache Ant](#) version 1.7 or later.
- [Apache Maven](#) 3.0 or later.
- The following environment variables must be set: `JAVA_HOME`, `JAVA5_HOME`, `FORREST_HOME`, and `ANT_HOME`.
- Your `PATH` must include the `JAVA_HOME`, `ANT_HOME`, `FORREST_HOME` and maven bin directories.
- If you are using Red Hat or CentOS systems, the `rpmdevtools` package is required for the `rpmdev-setuptree` command used below.

### Setting up an environment for building RPMs

#### Red Hat or CentOS systems

Users of these systems can run the following command to set up their environment:

```
$ rpmdev-setuptree                                # Creates ~/rpmbuild and
~/rpmmacros
```

#### SLES systems

Users of these systems can run the following command to set up their environment:

```
$ mkdir -p ~/rpmbuild/{BUILD,RPMS,S{OURCE,PEC,RPM}S}
$ echo "%_topdir $HOME/rpmbuild"> ~/.rpmmacros
```

### Building an RPM

Download SRPMs from [archive.cloudera.com](http://archive.cloudera.com). The source RPMs for CDH4 reside at [http://archive.cloudera.com/cdh4/redhat/5/x86\\_64/cdh/4/SRPMS/](http://archive.cloudera.com/cdh4/redhat/5/x86_64/cdh/4/SRPMS/), [http://archive.cloudera.com/cdh4/sles/11/x86\\_64/cdh/4/SRPMS/](http://archive.cloudera.com/cdh4/sles/11/x86_64/cdh/4/SRPMS/) or [http://archive.cloudera.com/cdh4/redhat/6/x86\\_64/cdh/4/SRPMS/](http://archive.cloudera.com/cdh4/redhat/6/x86_64/cdh/4/SRPMS/). Run the following commands as a non-root user, substituting the particular SRPM that you intend to build:

```
$ export SRPM=hadoop-0.20-0.20.2+320-1.src.rpm
$ rpmbuild --nodeps --rebuild $SRPM                # Builds the native RPMs
$ rpmbuild --nodeps --rebuild --target noarch $SRPM # Builds the java RPMs
```

The built packages can be found in `$HOME/rpmbuild/RPMS`.





## Getting Support

This section describes how to get support for CDH4:

- [Cloudera Support](#)
- [Community Support](#)
- [Report Issues](#)
- [Get Announcements about New Releases](#)

## Cloudera Support

Cloudera can help you install, configure, optimize, tune, and run Hadoop for large scale data processing and analysis. Cloudera supports Hadoop whether you run our distribution on servers in your own data center, or on hosted infrastructure services such as Amazon EC2, Rackspace, SoftLayer, or VMware's vCloud.

If you are a Cloudera customer, you can:

- Create a [Cloudera Support Ticket](#).
- Visit the [Cloudera Knowledge Base](#).
- Learn how to register for an account to create a support ticket at the [support site](#).

If you are not a Cloudera customer, learn how [Cloudera](#) can help you.

## Community Support

Register for the [Cloudera Users groups](#).

If you have any questions or comments about CDH, you can send a message to the CDH user's list: [cdh-user@cloudera.org](mailto:cdh-user@cloudera.org)

If you have any questions or comments about using Cloudera Manager, you can send a message to the Cloudera Manager user's list: [scm-users@cloudera.org](mailto:scm-users@cloudera.org)

## Report Issues

Cloudera tracks software and documentation bugs and enhancement requests for CDH on [issues.cloudera.org](https://issues.cloudera.org). Your input is appreciated, but before filing a request, please search the Cloudera issue tracker and "[CDH Manual Installation](#)" [Community Forum](#) for existing issues, and send a message to the CDH user's list, [cdh-user@cloudera.org](mailto:cdh-user@cloudera.org), or the CDH developer's list, [cdh-dev@cloudera.org](mailto:cdh-dev@cloudera.org).

If you would like to report or view software issues and enhancement requests for Cloudera Manager, visit this site: <https://issues.cloudera.org/browse/CM>.

## Get Announcements about New CDH and Cloudera Manager Releases

To get information about releases and updates for all products, sign up for [Release Announcements](#) forum.



## Apache and Third-Party Licenses

This section describes the licenses that apply to CDH4:

- [Apache](#)
- [Third-Party](#)

### Apache License

All software developed by Cloudera for CDH is released with an Apache 2.0 license. Please let us know if you find any file that doesn't explicitly state the Apache license at the top and we'll immediately fix it.

Apache License Version 2.0, January 2004 <http://www.apache.org/licenses/>

Copyright 2010-2013 Cloudera

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

<http://www.cloudera.com/content/cloudera-content/cloudera-docs/Licenses/Third-Party-Licenses/Third-Party-Licenses.html>.

