Hortonworks Data Platform

Reference

(Jun 12, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Reference

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper, and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

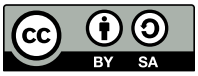


Table of Contents

List of Tables

1. Hadoop Service Accounts

This topic provides information about the service users for Hadoop.

HDP creates the following service users:

Note

You must always execute the `createUsers.sh` script file to ensure that these users are created by the installer.

The user names for these service users cannot be modified.

Table 1.1. Hadoop Service Accounts

HDP Service	User Name	Notes
HDFS	`hdfs`	NameNode, Secondary NameNode, and the DataNodes run as the `hdfs` user.
MapReduce	`mapred`	JobTracker, Job HistoryServer, and the TaskTrackers run as the `mapred` user.
HBase	`hbase`	HBase Master and the RegionServers run as the `hbase` user.
Hive	`hive`	Hive Metastore and HiveServer2 run as the `hive` user.
HCatalog	`hcat`	HCatalog runs as the `hcat` user. WebHCat Server also runs as the `hcat` user.
ZooKeeper	`zookeeper`	ZooKeeper server runs as the `zookeeper` user.
Oozie	`oozie`	Oozie server runs as the `oozie` user.

2. Configuring Ports

The tables below specify which ports must be opened for which ecosystem components to communicate with each other. Make sure the appropriate ports are opened before you install HDP.

HDFS Ports: The following table lists the default ports used by the various HDFS services.

Table 2.1. HDFS Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
NameNode WebUI	Master Nodes (NameNode and any back-up NameNodes)	50070	http	Web UI to look at current status of HDFS, explore file system	Yes (Typically admins, Dev/ Support teams)	`dfs.http.address`
		50470	https	Secure http service		`dfs.https.address`
NameNode metadata service	Master Nodes (NameNode and any back-up NameNodes)	8020/9000	IPC	File system metadata operations	Yes (All clients who directly need to interact with the HDFS)	Embedded in URI specified by `fs.default.name`
DataNode	All Slave Nodes	50075	http	DataNode WebUI to access the status, logs etc.	Yes (Typically admins, Dev/ Support teams)	`dfs.datanode.http.address`
		50475	https	Secure http service		`dfs.datanode.https.address`
		50010		Data transfer		`dfs.datanode.address`
		50020	IPC	Metadata operations	No	`dfs.datanode.ipc.address`
Secondary NameNode	Secondary NameNode and any backup Secondanry NameNode	50090	http	Checkpoint for NameNode metadata	No	`dfs.secondary.http.address`

MapReduce Ports: The following table lists the default ports used by the various MapReduce services.

Table 2.2. MapReduce Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
JobTracker WebUI	Master Nodes (JobTracker Node and any back-up Job-Tracker node)	50030	http	Web UI for JobTracker	Yes	`mapred.job.tracker.http.address`
JobTracker	Master Nodes (JobTracker Node)	8021	IPC	For job submissions	Yes (All clients who need to submit the MapReduce jobs including Hive, Hive server, Pig)	Embedded in URI specified by `mapred.job.tracker`
Task-Tracker Web UI and Shuffle	All Slave Nodes	50060	http	DataNode Web UI to access status, logs, etc.	Yes (Typically admins, Dev/ Support teams)	`mapred.task.tracker.http.address`
History Server WebUI		51111	http	Web UI for Job History	Yes	`mapreduce.history.server.http.address`

Hive Ports: The following table lists the default ports used by the various Hive services.

Note

Neither of these services is used in a standard HDP installation.

Table 2.3. Hive Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
Hive Server2	Hive Server machine (Usually a utility machine)	10000	thrift	Service for programatically (Thrift/ JDBC) connecting to Hive	Yes (Clients who need to connect to Hive either programatically or through UI SQL tools that use JDBC)	`ENV Variable HIVE_PORT`
Hive Metastore		9083	thrift	Service for accesing metadata about Hive	Yes (Clients that run Hive, Pig and	`hive.metastore.uris`

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
				tables and partitions.*	potentially M/R jobs that use HCatalog)	

* To change the metastore port, use this `hive` command: `hive --service metastore -p port_number`

HBase Ports: The following table lists the default ports used by the various HBase services.

Table 2.4. HBase Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
HMaster	Master Nodes (HBase Master Node and any back-up HBase Master node)	60000			Yes	`hbase.master.port`
HMaster Info Web UI	Master Nodes (HBase master Node and back up HBase Master node if any)	60010	http	The port for the HBase-Master web UI. Set to -1 if you do not want the info server to run.	Yes	`hbase.master.info.port`
Region Server	All Slave Nodes	60020			Yes (Typically admins, dev/ support teams)	`hbase.regionserver.port`
Region Server	All Slave Nodes	60030	http		Yes (Typically admins, dev/ support teams)	`hbase.regionserver.info.port`
	All ZooKeeper Nodes	2888		Port used by ZooKeeper peers to talk to each other.See here for more information.	No	`hbase.zookeeper.peerport`
	All ZooKeeper Nodes	3888		Port used by ZooKeeper		`hbase.zookeeper.leaderport`

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
				peers to talk to each other.See here for more information.		
		2181		Property from ZooKeeper's config `zoo.cfg`. The port at which the clients will connect.		`hbase.zookeeper.property.clientPort`

WebHCat Port: The following table lists the default ports used by the WebHCat service.

Table 2.5. WebHCat Port

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
WebHCat Server	Any utility machine	50111	http	Web API on top of HCatalog and other Hadoop services	Yes	`templeton.port`

Ganglia Ports: The following table lists the default ports used by the various Ganglia services.

Table 2.6. Ganglia Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
	Ganglia server	8660/61/62/63		For gmond collectors		
	All Slave Nodes	8660		For gmond agents		
	Ganglia server	8651		For ganglia gmetad		

MySQL Port: The following table lists the default port used by the MySQL service.

Table 2.7. MySQL Port

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
MySQL	MySQL database server	3306				

3. Controlling HDP Services Manually

Use the following instructions to start and/or stop HDP services manually:

- Starting HDP services
- Stopping HDP services

3.1. Starting HDP Services

Start all the Hadoop services in the following order:

- HDFS
- MapReduce
- ZooKeeper
- HBase
- Hive Metastore
- HiveServer2
- WebHCat
- Oozie
- Ganglia
- Nagios

Instructions

1. Start HDFS

   a. Execute these commands on the NameNode host machine:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf start namenode"
   ```

   b. Execute these commands on the Secondary NameNode host machine:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf start secondarynamenode"
   ```

   c. Execute these commands on all DataNodes:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf start datanode"
   ```

   where *$HDFS_USER* is the HDFS Service user. For example, `hdfs`.

2. Start MapReduce

   a. Execute these commands on the JobTracker host machine:

   ```
   su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
   etc/hadoop/conf start jobtracker; sleep 25"
   ```

b. Execute these commands on the JobTracker host machine:

```
su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
etc/hadoop/conf start historyserver"
```

c. Execute these commands on all TaskTrackers:

```
su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
etc/hadoop/conf start tasktracker"
```

where *$MAPRED_USER* is the MapReduce Service user. For example, `mapred`.

3. Start ZooKeeper. On the ZooKeeper host machine, execute the following command:

```
su - $ZOOKEEPER_USER -c "export  ZOOCFGDIR=/etc/zookeeper/conf ; export
 ZOOCFG=zoo.cfg ; source /etc/zookeeper/conf/zookeeper-env.sh ; /usr/lib/
zookeeper/bin/zkServer.sh start"
```

where *$ZOOKEEPER_USER* is the ZooKeeper Service user. For example, `zookeeper`.

4. Start HBase

a. Execute these commands on the HBase Master host machine:

```
su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
hbase/conf start master"
```

b. Execute these commands on all RegionServers:

```
su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
hbase/conf start regionserver"
```

where *$HBASE_USER* is the HBase Service user. For example, `hbase`.

5. Start Hive Metastore. On the Hive Metastore host machine, execute the following command:

```
 su -l $HIVE_USER -c "nohup hive --service metastore > $HIVE_LOG_DIR/hive.
out 2> $HIVE_LOG_DIR/hive.log   &"
```

where:

- *$HIVE_USER* is the Hive Service user. For example, `hive`.
- *$HIVE_LOG_DIR* is the directory where Hive server logs are stored (example:`/var/log/hive`).

6. Start HiveServer2. On the Hive Server2 host machine, execute the following command:

```
sudo su $HIVE_USER -c "nohup /usr/lib/hive/bin/hiveserver2 -hiveconf hive.
metastore.uris=\" \" > $HIVE_LOG_DIR /hiveServer2.out 2>$HIVE_LOG_DIR/
hiveServer2.log &"
```

This will start both the Hive Metastore and HCatalog services.

- *$HIVE_USER* is the Hive Service user. For example, `hive`.
- *$HIVE_LOG_DIR* is the directory where Hive server logs are stored (example:`/var/log/hive`).

7. Start WebHCat. On the WebHCat host machine, execute the following command:

```
su -l $WEBHCAT_USER -c "/usr/lib/hcatalog/sbin/webhcat_server.sh  start"
```

   where *$WEBHCAT_USER* is the WebHCat Service user. For example, `hcat`.

8. Start Oozie. On the Oozie server host machine, execute the following command:

```
 sudo su -l $OOZIE_USER -c "cd $OOZIE_LOG_DIR/log; /usr/lib/oozie/bin/oozie-start.sh"
```

   where:

   - *$OOZIE_USER* is the Oozie Service user. For example, `oozie`
   - *$OOZIE_LOG_DIR* is the directory where Oozie log files are stored (for example: `/var/log/oozie`).

9. Start Ganglia.

   a. Execute this command on the Ganglia server host machine:

```
/etc/init.d/hdp-gmetad start
```

   b. Execute this command on all the nodes in your Hadoop cluster:

```
/etc/init.d/hdp-gmond start
```

10 Start Nagios.

```
service nagios start
```

3.2. Stopping HDP Services

Before trying any upgrades or uninstalling software, stop all Hadoop services in the following order:

- Nagios
- Ganglia
- Oozie
- WebHCat
- Hive Metastore
- ZooKeeper
- HBase
- MapReduce
- HDFS

1. Stop Nagios. On the Nagios host machine, execute the following command:

```
service nagios stop
```

2. Stop Ganglia.

   a. Execute this command on the Ganglia server host machine:

   ```
   /etc/init.d/hdp-gmetad stop
   ```

   b. Execute this command on all the nodes in your Hadoop cluster:

   ```
   /etc/init.d/hdp-gmond stop
   ```

3. Stop Oozie. On the Oozie server host machine, execute the following command:

   ```
    sudo su -l oozie -c "cd $OOZIE_LOG_DIR/log; /usr/lib/oozie/bin/oozie-stop.
   sh"
   ```

   where:

   - *$OOZIE_USER* is the Oozie Service user. For example, `oozie`
   - *$OOZIE_LOG_DIR* is the directory where Oozie log files are stored (for example: `/var/log/oozie`).

4. Stop WebHCat. On the WebHCat host machine, execute the following command:

   ```
   su -l $WEBHCAT_USER -c "/usr/lib/hcatalog/sbin/webhcat_server.sh  stop"
   ```

   where:

   - *$WEBHCAT_USER* is the WebHCat Service user. For example, `hcat`.

5. Stop Hive. On the Hive Metastore host machine and Hive Server2 host machine, execute the following command:

   ```
   ps aux | awk '{print $1,$2}' | grep hive | awk '{print $2}' | xargs kill >/
   dev/null 2>&1
   ```

   This will stop Hive Metastore and HCatalog services.

6. Stop ZooKeeper. On the ZooKeeper host machine, execute the following command:

   ```
   su - $ZOOKEEPER_USER -c "export ZOOCFGDIR=/etc/zookeeper/conf ; export
    ZOOCFG=zoo.cfg ;source /etc/zookeeper/conf/zookeeper-env.sh ; /usr/lib/
   zookeeper/bin/zkServer.sh stop"
   ```

   where *$ZOOKEEPER_USER* is the ZooKeeper Service user. For example, `zookeeper`.

7. Stop HBase.

   a. Execute these commands on all RegionServers:

   ```
   su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf stop regionserver"
   ```

   b. Execute these commands on the HBase Master host machine:

   ```
   su -l $HBASE_USER -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/
   hbase/conf stop master"
   ```

where *$HBASE_USER* is the HBase Service user. For example, `hbase`.

8. Stop MapReduce

   a. Execute these commands on all TaskTrackers:

   ```
   su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
   etc/hadoop/conf stop tasktracker"
   ```

   b. Execute these commands on the JobTracker host machine:

   ```
   su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
   etc/hadoop/conf stop historyserver"
   ```

   c. Execute theses commands on the JobTracker host machine:

   ```
   su -l $MAPRED_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /
   etc/hadoop/conf stop jobtracker"
   ```

   where *$MAPRED_USER* is the MapReduce Service user. For example, `mapred`.

9. Stop HDFS

   a. Execute these commands on all DataNodes:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf stop datanode"
   ```

   b. Execute these commands on the Secondary NameNode host machine:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf stop secondarynamenode”
   ```

   c. Execute these commands on the NameNode host machine:

   ```
   su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh --config /etc/
   hadoop/conf stop namenode"
   ```

   where *$HDFS_USER* is the HDFS Service user. For example, `hdfs`.

4. Deploying HDP In Production Data Centers with Firewalls

This section describes mechanisms for deploying HDP in situations where a connection to the Internet is not possible or desirable. In this document:

- Deployment strategies for data centers with firewalls
- Recommendations for deploying HDP
- Detailed instructions for creating mirrors and proxies

4.1. Deployment Strategies for Data Centers with Firewalls

A typical Hortonworks Data Platform (HDP) install requires access to the Internet in order to fetch software packages from a remote repository. Since corporate networks typically have various levels of firewalls, these firewalls may limit or restrict Internet access, making it impossible for your cluster nodes to access the HDP repository during the install process.

The solution for this is to either:

- Create a local mirror repository inside your firewall hosted on a local mirror server inside your firewall; or
- Provide a trusted proxy server inside your firewall that can access the hosted repositories.

This document will cover these two options in detail, discuss the trade-offs, provide configuration guidelines, and will also provide recommendations for your deployment strategy.

In general, before installing Hortonworks Data Platform in a production data center, it is best to ensure that both the Data Center Security team and the Data Center Networking team are informed and engaged to assist with these aspects of the deployment.

4.1.1. Terminology

Table 4.1. Terminology

Item	Description
Yum Package Manager (yum)	A package management tool that fetches and installs software packages and performs automatic dependency resolution. See http://yum.baseurl.org/ or http://en.opensuse.org/Portal:Zypper for more information.
Local Mirror Repository	The Yum or Zypper repository hosted on your Local Mirror Server that will serve the HDP software.
Local Mirror Server	The server in your network that will host the Local Mirror Repository. This server must be accessible from all hosts in your cluster where you will install HDP.

Item	Description
HDP Repositories	A set of repositories hosted by Hortonworks that contains the HDP software packages. HDP software packages include the HDP Repository, HDP-UTILS Repository, and the Ambari Repository.
HDP Repository Tarball	A tarball image that contains the complete contents of the HDP Repositories.

4.1.2. Options for Mirroring or Proxying

HDP uses Yum or Zypper to install software, and this software is obtained from the: HDP Repositories, and the Extra Packages for Enterprise Linux (EPEL) repository. EPEL repository is used for RHEL/CentOS platforms.

If your firewall prevents Internet access, it will be necessary to mirror and/or proxy both the HDP repository and the Extra Packages for Enterprise Linux (EPEL) repository. Many Data Centers already mirror or proxy the EPEL repository, so discuss with your Data Center team whether EPEL is already available from within your firewall.

Mirroring a repository involves copying the entire repository and all its contents onto a local server and enabling an HTTPD service on that server to serve the repository locally. Once the local mirror server setup is complete, the `*.repo` configuration files on every repository client (i.e. cluster nodes) must be updated, so that the given package names are associated with the local mirror server instead of the remote repository server.

There are three options for creating a local mirror server. Each of these options is explained in detail in a later section.

- **Option I:** Mirror server has no access to Internet at all

  Use a web browser on your workstation to download the HDP Repository Tarball, move the tarball to the selected mirror server using scp or an USB drive, and extract it to create the repository on the local mirror server.

- **Option II:** Mirror server has temporary access to Internet

  Temporarily configure a server to have Internet access, download a copy of the HDP Repository to this server using the **reposync** command, then reconfigure the server so that it is back behind the firewall.

- **Option III:** Mirror server has permanent access to Internet (modified form of Option II)

  Establish a "trusted host", by permanently configuring a server to have Internet access, but still be accessible from within the firewall. Download a copy of the HDP Repository to this server using the **reposync** command.

Note

Option I is probably the least effort, and in some respects, is the most secure deployment option.

Option III is best if you want to be able to update your Hadoop installation periodically from the Hortonworks Repositories.

However, if you are considering Option III, you should also consider the fourth option, which is to proxy the HDP Repositories through a trusted proxy server. If you have a network administrator who has expertise in setting up proxies, and if the proxy option is acceptable within your Data Center Security policies, this can be the easiest of all the options.

- **Option IV:** Trusted proxy server

  Proxying a repository involves setting up a standard HTTP proxy on a local server to forward repository access requests to the remote repository server and route responses back to the original requestor. Effectively, the proxy server makes the repository server accessible to all clients, by acting as an intermediary.

  Once the proxy is configured, change the `/etc/yum.conf` file on every repository client (i.e. cluster nodes), so that when the client attempts to access the repository during installation, the request will go through the local proxy server instead of going directly to the remote repository server.

4.1.3. Considerations for choosing a Mirror or Proxy solution

The following table lists some benefits provided by these alternative deployment strategies:

Table 4.2. Comparison - HDP Deployment Strategies

Advantages of repository mirroring (Options I, II, and III)	Advantages of creating a proxy (Options IV)
• Minimizes network access (after the initial investment of copying the repository to local storage). The install process is therefore faster, reliable, and more cost effective (reduced WAN bandwidth minimizes the data center costs). • Allows security-conscious data centers to qualify a fixed set of repository files. It also ensures that the remote server will not change these repository files. • Large data centers may already have existing repository mirror servers for the purpose of OS upgrades and software maintenance. You can easily add the HDP Repositories to these existing servers.	• Avoids the need for long term management of the repository files (including periodic updates for upgrades, new versions, and bug fixes). • Almost all data centers already have a setup of well-known proxies. In such cases, you can simply add the local proxy server to the existing proxies' configurations. This approach is easier compared to creating local mirror servers in data centers with no mirror server setup. • The network access is same as that required when using a mirror repository, but the source repository handles file management.

However, each of the above approaches are also known to have the following disadvantages:

- Mirrors have to be managed for updates, upgrades, new versions, and bug fixes.
- Proxy servers rely on the repository provider to not change the underlying files without notice.
- Caching proxies are necessary, because non-caching proxies do not decrease WAN traffic and do not speed up the install process.

4.2. Recommendations for Deploying HDP

This section provides information on the various components of the Apache Hadoop ecosystem.

In many data centers, the following deployment strategy may be optimal:

- Use a mirror for the HDP Repositories. The HDP Repositories are small and easily mirrored, thereby allowing secure control over the contents of the Hadoop packages accepted for use in your data center.
- Use a caching proxy for the EPEL repository, which is a well-known and trustworthy repository managed by the Fedora Project team, and which may be too large to mirror in your data center. If your data center already mirrors or proxies EPEL, use that mirror or proxy.

Note

The installer pulls many packages from the base OS repositories (repos). If you do not have a complete base OS available to all your machines at the time of installation, you may run into issues. For example, if you are using RHEL 6 your hosts must be able to access the "Red Hat Enterprise Linux Server 6 Optional (RPMs)" repo. If this repo is disabled, the installation is unable to access the rubygems package.

If you encounter problems with base OS repos being unavailable, please contact your system administrator to arrange for these additional repos to be proxied or mirrored.

4.2.1. RPMs in the HDP repository

In the HDP repository, you will find two differenct source RPM for each component.

For example, for Hadoop, you should find the following two RPMs:

- `hadoop-x.x.x.x.el6.src.rpm`
- `hadoop-source-x.x.x.x.el6.i386.rpm`

The `src` and `source` are two different packages that serve the following purpose:

- The `src` package is used to re-create the binary in a given environment. You can use the `src` package of a particular component if you want to rebuild RPM for that component.
- The `source` package on the other hand, is used for reference or debuging purpose. The `source` package is particularly useful when you want to examine the source code of a particular component in a deployed cluster.

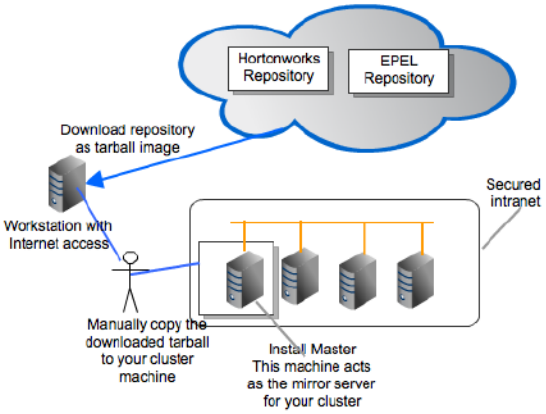
4.3. Detailed Instructions for Creating Mirrors and Proxies

In this section:

- Option III - Mirror server has permanent access to the Internet
- Option IV - Trusted proxy server

4.3.1. Option I - Mirror server has no access to the Internet

The local mirror setup for Option I is shown in the following illustration:



See the following instructions on configuring mirror server with no Internet access:

- Prerequisites
- Instructions

4.3.1.1. Prerequisites

To configure local repositories for HDP deployment, your system must meet the following minimum requirements:

- Your mirror server host must run on one of the following operating systems:
  - 64 bit Red Hat Enterprise Linux (RHEL)
  - 64 bit CentOS v5.x, v6.x
  - 64 bit SUSE Linux Enterprise Server (SLES) 11 SP1

  Ensure that this server and the cluster nodes are all running the same OS.

  Note

  To support repository mirroring for heterogeneous clusters requires a more complex procedure than the one documented here.
- The mirror server host must have several GB of storage available.
- The firewall allows all cluster nodes (the servers on which you want to install HDP) to access this server.
- If your mirror host uses SLES, execute the following command to install the required packages:

```
zypper -n --no-gpg-checks install apache2 apache2-prefork apache2-utils
 python-curl python-gobject2 python-gpgme python-urlgrabber python-iniparse
 wget curl yum-metadata-parser yum yum-updatesd yum-utils createrepo
```

4.3.1.2. Instructions

1. Use a workstation with access to the Internet and download the tarball image of the appropriate Hortonworks yum repository.

Table 4.3. Deploying HDP - Option I

Cluster OS	HDP Repository Tarballs
RHEL/ CentOS 5.x	• **HDP Repository:** `wget http://public-repo-1.hortonworks.com/HDP/centos5/HDP-1.3.0.0-centos5-rpm.tar.gz` • **HDP-Utils Repository:** `wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.15/repos/centos5/HDP-UTILS-1.1.0.15-centos5-rpm.tar.gz` • **Ambari Repository (Optional):** `wget http://public-repo-1.hortonworks.com/ambari/centos5/ambari-1.2.4.9-centos5.tar.gz`
RHEL/ CentOS 6.x	• **HDP Repository:** `wget http://public-repo-1.hortonworks.com/HDP/centos6/HDP-1.3.0.0-centos6-rpm.tar.gz` • **HDP-Utils Repository:** `wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.15/repos/centos6/HDP-UTILS-1.1.0.15-centos6-rpm.tar.gz` • **Ambari Repository (Optional):** `wget http://public-repo-1.hortonworks.com/ambari/centos6/ambari-1.2.4.9-centos6.tar.gz`
SLES 11	• **HDP Repository:** `wget http://public-repo-1.hortonworks.com/HDP/suse11/HDP-1.3.0.0-suse11-rpm.tar.gz` • **HDP-Utils Repository:** `wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.15/repos/suse11/HDP-UTILS-1.1.0.15-suse11-rpm.tar.gz` • **Ambari Repository (Optional):** `wget http://public-repo-1.hortonworks.com/ambari/suse11/ambari-1.2.4.9-suse11.tar.gz`

Note

The EPEL repository is not available as a tarball currently. Use one of Options II through IV to provide access to the EPEL repository.

2. Create an HTTP server.

   - On the mirror server, install an HTTP server (such as Apache httpd) using the instructions provided here.
   - Activate this web server.
   - Ensure that the firewall settings (if any) allow inbound HTTP access from your cluster nodes to your mirror server.

Note

If you are using EC2, make sure that SELinux is disabled.

3. On your mirror server, create a directory for your web server.
   - For example, from a shell window, type:
     - **For RHEL/CentOS:** `mkdir -p /var/www/html/hdp/`
     - **For SLES:** `mkdir -p /srv/www/htdocs/rpms`
   - If you are using a symlink, enable the `followsymlinks` on your web server.
4. Copy the HDP Repository Tarballs to the directory created in step 3, and untar these tarballs.
5. Verify the configuration.
   - The configuration is successful, if you can access the above directory through your web browser.

     To test this out, browse to the following URLs:
     - **HDP Repository:**

       ```
       http://$yourwebserver/hdp/HDP/$os/1.x/GA/1.3.0.0
       ```
     - **HDP-Utils Repository (Optional):**

       ```
       http://yourwebserver/hdp/HDP-UTILS-1.1.0.15/repos/$os
       ```
     - **Ambari Repository (Optional):**

       ```
       http://$yourwebserver/hdp/ambari/$os/1.x/updates/1.2.4.9
       ```

     where, *$os* can be centos5, centos6, or suse11.

     For each repository, you should see directory listing for the HDP or HDP-UTILS, or Ambari components along with the RPMs.
6. Configure the `yum` or `zypper` clients on all the nodes in your cluster.
   a. Fetch the yum configuration file from your mirror server.
     - **HDP Repository:**

       ```
       http://$yourwebserver/hdp/HDP/$os/1.x/GA/1.3.0.0/hdp.repo
       ```
     - **HDP-Utils Repository (Optional):**

       ```
       http://yourwebserver/hdp/HDP-UTILS-1.1.0.15/repos/$os/hdp-util.repo
       ```
     - **Ambari Repository (Optional):**

       ```
       http://$yourwebserver/hdp/ambari/$os/1.x/updates/1.2.4.9/ambari.repo
       ```

where, *$os* can be centos5, centos6, or suse11.

b. Store the repo file(s) in a temporary location.

c. Edit the repo file(s), changing the value of the **baseurl** property to the local mirror URL.

- **HDP Repository:** Edit the `hdp.repo` file changing the **baseurl** property as shown below:

```
[HDP-1.3.0.0]
name=Hortonworks Data Platform Version - HDP-1.3.0.0
baseurl=http://$yourwebserver/HDP/$os/1.x/GA/1.3.0.0
gpgcheck=1
gpgkey=http://$yourwebserver/HDP/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

[HDP-UTILS-1.1.0.15]
name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.15
baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.15/repos/$os
gpgcheck=1
gpgkey=http://$yourwebserver/HDP/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

- **HDP Utils Repository:** Edit the `hdp-utils.repo` file changing the **baseurl** property as shown below:

```
[HDP-UTILS-1.1.0.15]
name=Hortonworks Data Platform Version - HDP-UTILS-1.1.0.15
baseurl= http://$yourwebserver/HDP-UTILS-1.1.0.15/repos/$os
gpgcheck=1
enabled=1
priority=1
```

- **Ambari Repository (Optional):**

```
[ambari-1.x]
name=Ambari 1.x
baseurl=http://$yourwebserver/hdp/ambari/$os/1.x/GA
gpgcheck=1
gpgkey=http://public-repo-1.hortonworks.com/ambari/$os/RPM-GPG-KEY/RPM-
GPG-KEY-Jenkins
enabled=1
priority=1

[HDP-UTILS-1.1.0.15]
name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.15
baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.15/repos/$os
gpgcheck=0
gpgkey=http://$yourwebserver/ambari/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

[Updates-ambari-1.2.4.9]
name=ambari-1.2.4.9 - Updates
baseurl=http://$yourwebserver/ambari/$os/1.x/updates/1.2.4.9
```

```
gpgcheck=1
gpgkey=http://$yourwebserver/ambari/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

where, *$os* can be centos5, centos6, or suse11.

d. Copy the yum/zypper client configuration file to all nodes in your cluster.

- **For RHEL and CentOS:** Use **scp** or **pdsh** to copy the client yum configuration file to `/etc/yum.repos.d/` directory on every node in the cluster.
- **For SLES:**

i. Store the repo file(s) back into the repository location in the web server.

- **HDP Repository:**

```
cp /tmp/hdp.repo /srv/www/htdocs/rpms/hdp/HDP/suse11/1.x/GA/1.3.0.0
```

- **HDP-Utils Repository (Optional):**

```
cp /tmp/hdp.repo /srv/www/htdocs/rpms/hdp/HDP-UTILS-1.1.0.15/
suse11/
```

- **Ambari Repository (Optional):**

```
cp /tmp/ambari.repo /srv/www/htdocs/rpms/hdp/ambari/suse11/1.x/
updates/1.2.4.9
```

ii. On every node, invoke the following command:

- **HDP Repository:**

```
zypper addrepo -r http://$yourwebserver/hdp/HDP/suse11/1.x/GA/1.3.
0.0/hdp.repo
```

- **HDP Repository:**

```
zypper addrepo -r http://$yourwebserver/hdp/HDP-UTILS-1.1.0.15/
suse11/hdp-util.repo
```

- **Ambari Repository (Optional):**

```
zypper addrepo -r http://$yourwebserver/hdp/ambari/suse11/1.x/
updates/1.2.4.9/ambari.repo
```

7. If your cluster runs CentOS or RHEL, and if you have multiple repositories configured in your environment, deploy the following plugin on all the nodes in your cluster.

a. Install the plugin.

- **For RHEL and CentOS v5.x**

```
yum install yum-priorities
```

- **For RHEL and CentOS v6.x**

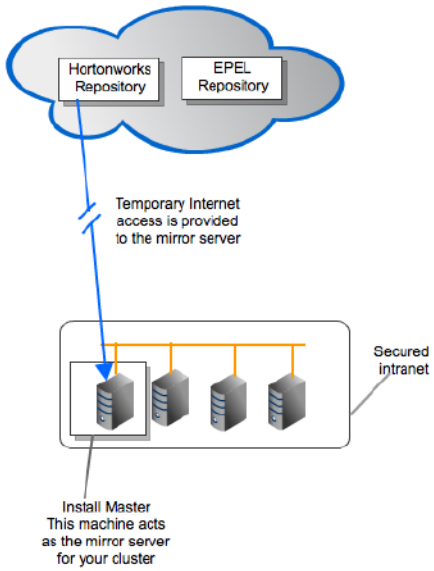
```
yum install yum-plugin-priorities
```

b. Edit the `/etc/yum/pluginconf.d/priorities.conf` file to add the following:

```
[main]
enabled=1
gpgcheck=0
```

4.3.2. Option II - Mirror server has temporary access to the Internet

The local mirror setup for Option II is shown in the following illustration:



See the following instructions for configuring mirror server that has temporary access to the Internet:

- Prerequisites
- Instructions

4.3.2.1. Prerequisites

To configure local repositories for HDP deployment, your system must meet the following minimum requirements:

- Your mirror server host must run on one of the following operating systems:
  - 64 bit Red Hat Enterprise Linux (RHEL)
  - 64 bit CentOS v5.x, v6.x
  - 64 bit SUSE Linux Enterprise Server (SLES) 11 SP1

  Ensure that this server and the cluster nodes are all running the same OS.

Note

To support repository mirroring for heterogeneous clusters requires a more complex procedure than the one documented here.

- The mirror server host must have several GB of storage available.
- The firewall allows all cluster nodes (the servers on which you want to install HDP) to access this server.
- Ensure that the mirror server has **yum** installed.
- Add the **yum-utils** and **createrepo** packages on the mirror server.

```
yum install yum-utils createrepo
```

- If your mirror host uses SLES, execute the following command to install the required packages:

```
zypper -n --no-gpg-checks install apache2 apache2-prefork apache2-utils
 python-curl python-gobject2 python-gpgme python-urlgrabber python-iniparse
 wget curl yum-metadata-parser yum yum-updatesd yum-utils createrepo
```

4.3.2.2. Instructions

1. Temporarily reconfigure your firewall to allow Internet access from your mirror server host.
2. Execute the following command to download the appropriate Hortonworks yum client configuration file and save it in `/etc/yum.repos.d/` directory on the mirror server host.

Table 4.4. Deploying HDP - Option II

Cluster OS	HDP Repository Tarballs

RHEL/ CentOS 5.x	• **HDP Repository:**
	`wget http://public-repo-1.hortonworks.com/HDP/centos5/1.x/GA/1.3.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo`
	• **Ambari Repository (Optional):**
	`wget http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/1.2.4.9/ambari.repo -O /etc/yum.repos.d/ambari.repo`

RHEL/ CentOS 6.x	• **HDP Repository:**
	`wget http://public-repo-1.hortonworks.com/HDP/centos6/1.x/GA/1.3.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo`
	• **Ambari Repository (Optional):**
	`wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.2.4.9/ambari.repo -O /etc/yum.repos.d/ambari.repo`

SLES 11	• **HDP Repository:** `wget http://public-repo-1.hortonworks.com/HDP/suse11/1.x/GA/1.3.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo` • **Ambari Repository (Optional):**

Cluster OS	HDP Repository Tarballs
	`wget http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.2.4.9/ambari.repo -O /etc/zypp/repos.d/ambari.repo`

Note

If you are using Ambari to perform the HDP installation, you will need to setup the Ambari repository using the information provided above.

3. Create an HTTP server.

   - On the mirror server, install an HTTP server (such as Apache `httpd`) using the instructions provided here.
   - Activate this web server.
   - Ensure that the firewall settings (if any) allow inbound HTTP access from your cluster nodes to your mirror server.

Note

If you are using EC2, make sure that SELinux is disabled.

   - [Optional]: If your mirror server uses SLES, modify the `default-server.conf` file to enable the docs root folder listing.

```
sed -e "s/Options None/Options Indexes MultiViews/ig" /etc/apache2/
default-server.conf > /tmp/tempfile.tmp
mv /tmp/tempfile.tmp /etc/apache2/default-server.conf
```

4. On your mirror server, create a directory for your web server.

   - For example, from a shell window, type:
     - **For RHEL/CentOS:** `mkdir -p` *`/var/www/html/hdp/`*
     - **For SLES:** `mkdir -p` *`/srv/www/htdocs/rpms`*
   - If you are using a symlink, enable the `followsymlinks` on your web server.

5. Copy the contents of entire HDP repository from the remote yum server to your local mirror server.

   Continuing the previous example, from a shell window, type:

   - Navigate to the directory created on your web server previously:
     - **For RHEL/CentOS:**

```
cd /var/www/html/hdp
```

     - **For SLES:**

```
cd /srv/www/htdocs/rpms
```

- Check version for `yum-utils` library.

```
rpm -q yum-utils
```

If the current version of `yum-utils` library is lesser than `1.1.31-7`, there is might be a potential problem while running `reposync` command due a bug. The workaround is available at https://bugzilla.redhat.com/show_bug.cgi?id=882536.

- Copy the contents of entire HDP repository from the remote yum server to your local mirror server.
  - **HDP Repository:**

```
reposync -r HDP
reposync -r Updates-HDP-1.x
reposync -r HDP-UTILS-1.1.0.15
```

  - **Ambari Repository (Optional):**

```
reposync –r ambari-1.x
reposync –r Updates-ambari-1.2.4.9
```

6. Generate appropriate metadata. This step defines each directory as a yum repository.

From a shell window, type:

- **For RHEL/CentOS:**
  - **HDP Repository:**

```
createrepo /var/www/html/hdp/HDP
```

  - **Ambari Repository (Optional):**

```
createrepo /var/www/html/hdp/ambari-1.x
createrepo /var/www/html/hdp/Updates-ambari-1.2.4.9
```

- **For SLES:**

HDP Repository:

```
createrepo /srv/www/htdocs/rpms/hdp/HDP
```

  - **Ambari Repository (Optional):**

```
createrepo /srv/www/htdocs/rpms/hdp/ambari-1.x
createrepo/srv/www/htdocs/rpms/hdp/Updates-ambari-1.2.4.9
```

You should see a new `repodata` directory under the HDP and Updates-HDP-1.x directories.

If using Ambari, you should also see the `repodata` directory under the `ambari-1.x` and `Updates-ambari-1.x` directories.

7. Verify the configuration.

- The configuration is successful, if you can access the above directory through your web browser.

To test this out, browse to the following URLs:

- **HDP Repository:**

```
http://$yourwebserver/hdp/HDP/$os/1.x/GA/1.3.0.0
```

- **Ambari Repository (Optional):**

```
http://$yourwebserver/hdp/ambari/$os/1.x/updates/1.2.4.9
```

where, *$os* can be centos5, centos6, or suse11.

- You should now see directory listing for all the HDP components.

8. At this point, it is okay to disable external Internet access for the mirror server, so that the mirror server is once again entirely within your data center firewall.

9. Configure the `yum` or `zypper` clients on all the nodes in your cluster.

   a. Edit the repo file(s), changing the value of the **baseurl** and **gpgkey** properties to the local mirror URL.

   - Edit the `/etc/yum.repos.d/hdp.repo` file changing the **baseurl** property as shown below:

```
[HDP-1.3.0.0]
name=Hortonworks Data Platform Version - HDP-1.3.0.0
baseurl=http://$yourwebserver/HDP/$os/1.x/GA/1.3.0.0
gpgcheck=1
gpgkey=http://$yourwebserver/HDP/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

[HDP-UTILS-1.1.0.15]
name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.15
baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.15/repos/$os
gpgcheck=1
gpgkey=http://$yourwebserver/HDP/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

   - Edit the `/etc/yum.repos.d/ambari.repo` file changing the **baseurl** property as shown below:

```
[ambari-1.x]
name=Ambari 1.x
baseurl=http://$yourwebserver/hdp/ambari/$os/1.x/GA/ambari.repo
gpgcheck=1
gpgkey=http://$yourwebserver/ambari/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1

[HDP-UTILS-1.1.0.15]
name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.15
baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.15/repos/$os
gpgcheck=0
gpgkey=http://$yourwebserver/ambari/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

```
[Updates-ambari-1.2.4.9]
name=ambari-1.2.4.9 - Updates
baseurl=http://$yourwebserver/ambari/$os/1.x/updates/1.2.4.9
gpgcheck=1
gpgkey=http://$yourwebserver/ambari/$os/RPM-GPG-KEY/RPM-GPG-KEY-Jenkins
enabled=1
priority=1
```

where, *$os* can be centos5, centos6, or suse11.

b. Copy the yum/zypper client configuration file to all nodes in your cluster.

- **For RHEL and CentOS:** Use **scp** or **pdsh** to copy the client yum configuration file to `/etc/yum.repos.d/` directory on every node in the cluster.
- **For SLES:** On every node, invoke the following command:
  - **HDP Repository:**

    ```
    zypper addrepo -r http://$yourwebserver/hdp/HDP/suse11/1.x/GA/1.3.0.0/hdp.repo
    ```

  - **Ambari Repository (Optional):**

    ```
    zypper addrepo -r http://$yourwebserver/hdp/ambari/suse11/1.x/updates/1.2.4.9/ambari.repo
    ```

- If using Ambari, verify the configuration by deploying Ambari server on one of the cluster nodes.

  ```
  yum install ambari-server
  ```

10. If your cluster runs CentOS or RHEL, and if you have multiple repositories configured in your environment, deploy the following plugin on all the nodes in your cluster.

a. Install the plugin.

- **For RHEL and CentOS v5.x**

  ```
  yum install yum-priorities
  ```

- **For RHEL and CentOS v6.x**

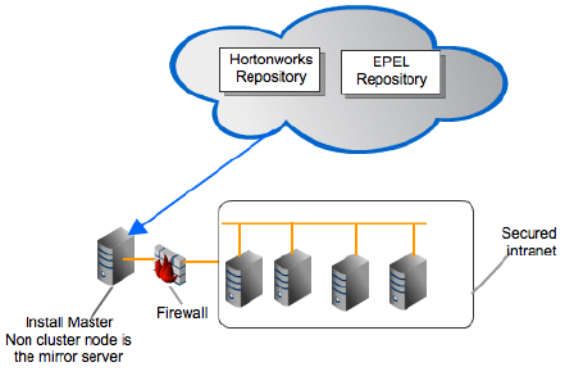
  ```
  yum install yum-plugin-priorities
  ```

b. Edit the `/etc/yum/pluginconf.d/priorities.conf` file to add the following:

```
[main]
enabled=1
gpgcheck=0
```

4.3.3. Option III - Mirror server has permanent access to the Internet

The local mirror setup for Option III is shown in the following illustration:



See the following instructions for configuring mirror server that has permanent access to the Internet:

- Prerequisites
- Instructions

4.3.3.1. Prerequisites

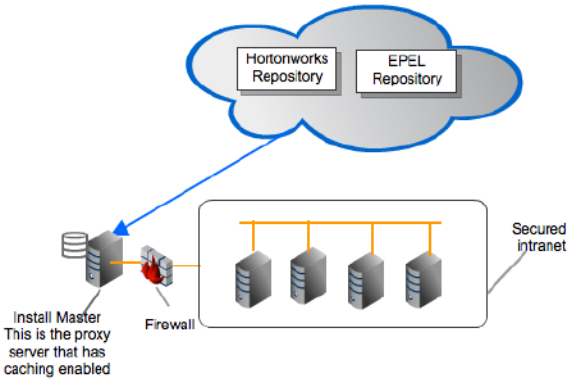
Same as Option II.

4.3.3.2. Instructions

Same as Option II but Step 1 and 8 are unnecessary and may be skipped.

4.3.4. Option IV - Trusted proxy server

The local mirror setup for Option IV is shown in the following illustration:



See the following instructions for configuring trusted proxy server:

- Prerequisites
- Instructions

4.3.4.1. Prerequisites

Select a mirror server host with the following characteristics:

- This server runs on either CentOS (v5.x, v6.x) or RHEL (v5.x, v6.x) and has several GB of storage available.
- The firewall allows all cluster nodes (the servers on which you want to install HDP) to access this server, and allows this server to access the Internet (at least those Internet servers for the repositories to be proxied).

4.3.4.2. Instructions

1. Create a caching HTTP PROXY server on the selected host.
   a. It is beyond the scope of this document to show how to set up an HTTP PROXY server, given the many variations that may be required, depending on your data center's network security policy. If you choose to use the Apache HTTPD server, it starts by installing **httpd**, using the instructions provided here, and then adding the **mod_proxy** and **mod_cache modules**, as stated here. Please engage your network security specialists to correctly set up the proxy server.
   b. Activate this proxy server and configure its cache storage location.
   c. Ensure that the firewall settings (if any) allow inbound HTTP access from your cluster nodes to your mirror server, and outbound access to the desired repo sites, including **public-repo-1.hortonworks.com**.

   Note

   If you are using EC2, make sure that SELinux is disabled.

2. Depending on your cluster OS, configure the `yum` or `zypper` clients on all the nodes in your cluster.
   - **For RHEL and CentOS:**

     Note

     The following description is taken from the CentOS documentation here.

     a. On each cluster node, add the following lines to the **/etc/yum.conf** file.

     (As an example, the settings below will enable **yum** to use the proxy server **mycache.mydomain.com**, connecting to port **3128**, with the following credentials **yum-user/qwerty**.

```
# proxy server:port number
proxy=http://mycache.mydomain.com:3128

# account details for secure yum proxy connections
proxy_username=yum-user
proxy_password=qwerty
```

b. Once all nodes have their **/etc/yum.conf** file updated with appropriate configuration info, you can proceed with the HDP installation just as though the nodes had direct access to the Internet repositories.

c. If this proxy configuration does not seem to work, try adding a **/** at the end of the proxy URL. For example:

```
proxy=http://mycache.mydomain.com:3128/
```

- **For SLES:**

a. Configure the zypper clients on all the nodes in your cluster, to use the proxy server.

b. Please consult with your system administrator to do this correctly in your environment.

5. Installing the JDK

HDP requires that the Java SE Development Kit (JDK) v1.6 update 31 or later must be installed on all the nodes in your cluster. Follow the instructions listed below to deploy the JDK manually:

1. Obtain the version of JDK currently installed on your host machine.

```
java -version
```

2. Uninstall the Java package if JDK version is less than v 1.6 update 31.

```
rpm -qa | grep java
yum remove java-1.x.0-jdk-1.x.0.0-1.45.1.11.1.el6.x86_64
```

Step 3: Verify that the default Java package is uninstalled.

```
which java
```

3. Download Oracle JDK jdk-6u31-linux-x64.binon all the host machines.

4. Change directory to the location where you downloaded the JDK and run the install.

```
mkdir /usr/jdk1.6.0_31
cd /usr/jdk1.6.0_31
chmod u+x $JDK_download_directory/jdk-6u31-linux-x64.
bin
$JDK_download_directory/jdk-6u31-linux-x64.bin -noregister
```

5. Create symbolic links (symlinks) to the JDK.

```
mkdir /usr/java
ln -s /usr/jdk1.6.0_31/jdk1.6.0_31 /usr/java/default
ln -s /usr/java/default/bin/java /usr/bin/java
```

6. Set up your environment to define JAVA_HOME to put the Java Virtual Machine and the Java compiler on your path.

```
export JAVA_HOME=/usr/java/default
export PATH=$JAVA_HOME/bin:$PATH
```

6. Supported Database Matrix for Hortonworks Data Platform

This page contains certification information on supported databases for Hortonworks Data Platform (HDP).

The following table identifies the supported databases for HDP.

Table 6.1. Supported Databases for HDP Stack

Operating System	Component	Database			
		PostgreSQL 8.x	**MySQL 5.x**	**Oracle 11gr2**	**Other**
RHEL 5.x, 6.x | CentOS 5.x, 6.x | SLES 11	Hive / HCatalog	Default. For instructions on configuring this database for Hive metastore, see: Installing HDP manually.	Default. For instructions on configuring this database for Hive metastore, either see Instructions for Manual Install or see Instructions for Ambari.	Supported. For instructions on configuring this database for Hive metastore, either see Instructions for Manual Install or see Instructions for Ambari.	
	Oozie	Supported. For instructions on configuring this database for Oozie metastore, see: Installing HDP manually.	Supported. For instructions on configuring this database for Oozie metastore, either see Instructions for Manual Install or see Instructions for Ambari.	Supported. For instructions on configuring this database for Oozie metastore, see: either Instructions for Manual Install or Instructions for Ambari.	Derby (default).

Table 6.2. Supported Databases for Ambari

Operating System	Component	Database
RHEL 5.x, 6.x | CentOS 5.x, 6.x | SLES 11	Ambari	PostgreSQL 8.x (default), Oracle 11gr2. For more information on configuring Oracle 11gr2 instance as Ambari database, see Appendix: Using Non-Default Databases - Ambari

7. Appendix: MySQL Replication for Failover Protection

This appendix explains how to set up MySQL replication using either a master-slave configuration or master-master in active-passive mode.

7.1. Replication Models

This document describes two alternative replication models that can be used when configuring MySQL for failover protection:

- Master-Slave (M-S)
- Master-Master in Active-Passive mode (M-M-A-P)

Important

Please consult with your internal DBAs. They already know how to do this. Setting up database replication in a reliable and highly available way requires accommodation to your specific application environment, and these instructions are meant to be a guide, not a complete specification guaranteed to be sufficient in all circumstances. This information is provided "AS IS" and is not warranteed by Hortonworks Inc.

7.1.1. Master-Slave Model

The Master-Slave model is the most basic replication model, where one box acts as a master, generating transaction records which are then replicated over to the slave. This creates a cold standby that can then be used as a backup or as a replacement for the master when it goes down. Manual configuration can also result in swapping around the order of the master and the slave.

- Pros: Robust, well-understood. Allows for backups, simple to configure and manage.
- Cons: Requires explicit manual intervention to switch it over when failure of the master happens.

7.1.1.1. Master-Slave Runtime Configurations

Table 7.1. M-S Runtime Configurations: Master

Configurations Set on Master	Comments
`server_id`	Change from default 1 to a unique id.
`log_bin`	On, set to a filename where all transactions are recorded for transfer to the slave.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
`innodb_flush_logs_at_trx_commit`	1 — recommended setting, default with newer MySQL versions.
`innodb_support_xa`	1 — recommended setting, default with newer MySQL versions.

Table 7.2. M-S Runtime Configurations: Slave

Configurations Set on Slave	Comments
`server_id`	Change from default 1 to a unique id separate from the master.
`read_only`	Turned on — prevents any application from accidentally attempting to write as a result of a misconfiguration.
`relay_log`	On, set to a filename to use.
`log_slave_updates`	Turned on — this is not strictly necessary and is mostly used for slave chains, but makes it much easier to recover and rebuild the original master after a failure; therefore using `log_slave_updates` is recommended.
`log_bin`	On — again not strictly necessary on the slave, but useful to have when rebuilding.
`binlog_format`	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
Master setting	Set to the ip/name of the master (done directly in MySQL, not in `my.cnf`).

7.1.2. Master-Master-Active-Passive Model

First of all, it should be noted that MySQL does not support what is often considered to be a multi-master model, in that no slave can receive replication records from more than one master. However, it is possible for us to set up two machines which each refer to each other as master, and as long as they each have only one master (each other), we can have multiple "masters". It is also possible to have a Master-chain setup, but that brings up more problems than it solves, so we'll restrict ourselves primarily to the 2-master model in this discussion, and compare it to a typical M-S setup.

While a Master-Master model does support, in theory, updates to both masters, it is recommended to not use it in that fashion, as it can lead to inconsistencies between the two masters in cases of updates involving auto-increment fields or updates that update the same field. Also, there isn't any major benefit to writing across two masters in terms of write scaling, since the update will have to be processed on both computers anyway. For the most part, the main advantage that an M-M model gives over an M-S model is simply the ability to switch over to a secondary master upon failure of the primary master and to continue using the application that uses the database, and eventually, when the original master comes back up, it can recover much more easily than in an M-S situation. Thus, for the purposes of using a MySQL database as a backend for the Hive metastore, it is recommended that the M-M model not be used in what's referred to as Active-Active mode, but rather, that it be used in an Active-Passive mode, where writes happen to only one master at any given time.

7.1.2.1. M-M-A-P Runtime Configurations

Table 7.3. M-M-A-P Runtime Configurations: Master-A

Configurations Set on Master-A	Comments
`server_id`	Change from default 1 to a unique id.
`read_only`	Turned off — this will be our "active" master.
`relay_log`	On, set to a filename to use.

Configurations Set on Master-A	Comments
log_slave_updates	Turned on.
log_bin	On, set to a filename to use.
binlog_format	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
innodb_flush_logs_at_trx_commit	1 — recommended setting, default with newer MySQL versions.
innodb_support_xa	1 — recommended setting, default with newer MySQL versions.
Master setting	Set to master-B (done directly in MySQL, not in my.cnf).

Table 7.4. M-M-A-P Runtime Configurations: Master-B

Configurations Set on Master-B	Comments
server_id	Change from default 1 to a unique id separate from the master-A.
read_only	Turned on — prevents any application from accidentally attempting to write as a result of a misconfiguration.
relay_log	On, set to a filename to use.
log_slave_updates	Turned on.
log_bin	On, set to a filename to use.
binlog_format	ROW. The default isolation level for the Hive metastore is repeatable-read, and bin logs are not supported in STATEMENT or MIXED modes with that setting.
innodb_flush_logs_at_trx_commit	1 — recommended setting, default with newer MySQL versions.
innodb_support_xa	1 — recommended setting, default with newer MySQL versions.
Master setting	Set to master-A (done directly in MySQL, not in my.cnf).

7.2. Configuring and Using Replication

This section explains how to configure MySQL replication with either the Master-Slave model (M-S) or the Master-Master model in Active-Passive mode (M-M-A-P), and describes how to recover from failures in each case.

7.2.1. Disclaimer and Assumptions

- A lot of the following instructions assume that you are using InnoDB for all your MySQL tables. If you are using some other storage engine, other changes might be required and it would be best to verify with a DBA. InnoDB is the recommended storage engine for storing Hive metadata.
- Assume we want to use server-A.my.example.com as the primary master and server-B.my.example.com as the slave (or secondary passive master in the M-M-A-P case). Let's say we have installed Hive so as to use server-A as its metastore database, as one would when installing with the HDP installer, and we've simply installed MySQL (using yum or otherwise) on server-B, but done nothing else.

7.2.2. M-S Replication

7.2.2.1. Setup of Master-Slave Replication

First, on server-A.my.example.com:

1. Shut down the mysql server process if it is running.

```
> mysqladmin shutdown
```

2. Edit the `my.cnf` files with the following values:

```
log_bin=mysql-bin
binlog_format=ROW
server_id=10
innodb_flush_logs_at_trx_commit=1
innodb_support_xa=1
```

3. Bring up the mysql server process again — this step may vary based on your installation; in a typical RHEL setup, I can use the system service startup for mysql as follows:

```
> service mysql start
```

4. To verify that the server is now logging bin-logs, you can use the SQL command: "`SHOW MASTER STATUS;`". It should show you a binlog filename and a position.

5. Make sure that your current user is able to do a dump of the MySQL database, by running the following as a mysql-root-capable user, for example, on a default installation, "root".

```
CREATE USER ‘root’@’server-A.my.example.com’ identified by ‘p4ssw0rd’;
GRANT ALL ON *.* to ‘root’@’server-A.my.example.com’;
```

   Also, create a replication user that will be used to conduct future replications:

```
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'server-B.my.
example.com'\
  IDENTIFIED BY 'r3plpwd';
```

6. Run `mysqldump` to dump all tables from the master and load them onto the slave as follows:

```
> mysqldump --single-transaction --all-databases --master-data=1 --host=
server-A.my.example.com > dump.out
```

   (You may need to specify `-pp4ssw0rd` to specify the password.)

7. Copy this dump.out file over to the server-B.

Then, on `server-B.my.example.com`:

1. Shut down the mysql server process if it is running.

2. Edit the my.cnf files with the following values:

```
log_bin = mysql-bin
binlog_format = ROW
server_id = 11
relay_log = mysql-relay-bin
log_slave_updates = 1
read_only = 1
```

3. Bring up the mysql server process.

4. Make sure that your current user is able to load the prepared dump of the MySQL database, by running the following as a mysql-root-capable user, for example, on a default installation, "root".

```
create user 'root'@'server-B.my.example.com' identified by 'p4ssw0rd';
grant all on *.* to 'root'@'server-B.my.example.com';
```

5. Load the dump that was dumped out by `mysqldump` by running the following:

```
> mysql --host=server-B.my.example.com -pp4ssw0rd < dump.out
```

6. Verify that the metastore database was transferred over by running a 'SHOW DATABASES' call on MySQL.

7. Look through the MySQL dump, and locate values for MASTER_LOG_FILE and MASTER_LOG_POS. We will need to specify values for these to start replication on the slave. Assuming these values were 'mysql-bin.000001' and the position was 0, then to copy new entries from the master, run the following:

```
CHANGE MASTER TO MASTER_HOST='server-A.my.example.com', MASTER_USER='repl',\
  MASTER_PASSWORD='r3plpwd', MASTER_LOG_FILE='mysql-bin.000001',
 MASTER_LOG_POS=0;
```

   Note that these values can also be obtained from the master by running 'SHOW MASTER STATUS' on it.

8. Restart the mysql server.

9. Check that the replication is correctly configured by running

```
SHOW SLAVE STATUS;
```

   or, for increased readability:

```
SHOW SLAVE STATUS\G
```

It should show correct values as set previously for Master_User and Master_Host. If the slave is caught up to the master, then this field will show a value for Seconds_Behind_Master as being 0.

And with that, you now have M-S replication set up.

7.2.2.2. What To Do in the Case of a Master Failure

1. Stop all writes on the existing master — for example, shut down the Hive metastore server and stop any Hive processes that connect directly to the database.
2. If the master is still accessible, you can flush the tables to make sure that everything has been written out to logs with the command 'FLUSH TABLES WITH READ LOCK' and set the master to read_only.
3. Make sure that replication in the slave is caught up to the transactions recorded by the master by checking 'SHOW SLAVE STATUS'.
4. Run 'STOP SLAVE' on the slave.
5. Remove existing master settings for replication by running:

```
CHANGE MASTER TO MASTER_HOST=''
```

6. Note the new master's binary log coordinates with 'SHOW MASTER STATUS'.
7. Configure a new slave to replicate from this new master in a manner similar to how we configured the M-S setup before.
8. Change hive-site.xml to point to the new master.
9. If you intend to bring back the old master in this rotation, it can easily be configured to be a slave of the new master.
10. Turn off the read-only flag on the new master (old slave).

7.2.3. M-M-A-P Replication

7.2.3.1. Configuration for Master-Master-Active-Passive

1. Follow the same M-S setup instructions as above, including log_bin, binlog_format and log_slave_updates as recommended. While these settings were not essential for a typical M-S setup, they are needed for an M-M-A-P setup.
2. Verify that both machines are in sync, that one-way replication is working.
3. Configure the server-A's master as server-B, in a manner similar to how we configured server-B's master as server-A. To wit, as follows:

   On server-B, after verifying sync:

```
```

```
GRANT REPLICATION SLAVE, REPLICATION CLIENT ON *.* TO 'repl'@'server-A.my.
example.com' IDENTIFIED BY 'r3plpwd';
SHOW MASTER STATUS;
```

Let's say, for example, that we see that the replication file is mysql-bin.000004 and the position is 296. Then, we go to server-A, and run the following:

```
CHANGE MASTER TO MASTER_HOST='server-B.my.example.com', MASTER_USER='repl',
 MASTER_PASSWORD='r3plpwd',\
MASTER_LOG_FILE='mysql-bin.000004',MASTER_LOG_POS=296;
START SLAVE;
```

That's all there is to it — you're set. If you still don't see a slave process running, you might need to restart the slave database as a means of ensuring all settings being flushed and loaded.

7.2.3.2. Switching Active and Passive Roles in the M-M-A-P Setup

This is a simple matter of doing the following:

1. Shut down the applications (metastore server) so as to not perform any new writes to the database.
2. Switch the "active" master to read-only.
3. Wait for the "passive" master to be caught up to replicating all updates from the old "active" master.
4. Switch the passive master's read-only flag to off.
5. Switch config in the applications (metastore server) to point to the old passive/new active master and restart the application.

That's it — this switches which master is active and which is passive.

7.2.3.3. Recovering from Failure When the Active Master in an M-M-A-P Fails

- If the old master still has not had any data corruption, it should recover gracefully when its replication from the new master is caught up.
- If there was data loss/data corruption, proceed to set up a database as if it were a fresh slave in the M-S case, and/or the secondary master in the M-M-A-P case, and continue.