Hortonworks Data Platform

Reference

(Jul 2, 2014)

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

You can configure service accounts using:

- If you are performing a Manual Install of HDP, refer to the Manual instructions on Service User and Group accounts
- If you are performing a Ambari Install of HDP, refer to the Ambari instructions in the Ambari Install Guide .

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.

Accumulo Ports

The following table lists the default ports used by the various Accumulo services. (**Note:** Neither of these services are used in a standard HDP installation.)

Table 2.1. Accumulo Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration
Master	Master nodes (Active master and any standby)	9999		The Master thrift server	Yes (client API needs)	`master.por`
TabletServer	Slave nodes	9997		The TabletServer thrift server	Yes (client API needs)	`tserver.po` site.xml
Garbage Collector	GC nodes (Active GC and any standby)	50091		The GarbageCollector thrift server	No	`gc.port.cl`
Monitor	Monitor nodes (Active Monitor and any standby)	50095	HTTP(S)	Metrics/Monitoring of an Accumulo instance	Yes	`monitor.po` site.xml
Monitor log aggregation	Monitor nodes (Active Monitor and any standby)	4560		Log4j socket which accepts logs forwarded from other Accumulo services	No	`monitor.po`
Tracer	Tracer nodes	12234		The Tracer thrift server	Yes (if enabled)	`trace.port`
Thrift Proxy (optional)	Proxy nodes	42424		The Thrift Proxy server	Yes (if enabled)	port in `proxy`
TabletServer Replication Service	Slave nodes	10002		TabletServer Thrift service supporting multi-instance Accumulo replication	No	`replicatio` in accumulo-si
Master Replication Service	Master nodes (Active master and any standby)	10001		Master Thrift service supporting multi-instance Accumulo replication	No	`master.rep` in accumulo-si

Flume Ports

The following table lists the default ports used by the various Flume services. (**Note:** Neither of these services are used in a standard HDP installation.)

Table 2.2. Flume Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration
Flume	Flume Agent	41414	TCP	Flume performance metrics in JSON format	Yes (client API needs)	`master.por`

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuratio
Flume	HDFS Sink	8020	TCP	Communication from Flume into the Hadoop cluster's NameNode	Yes (client API needs)	`tserver.po`
Flume	HDFS Sink	9000	TCP	Communication from Flume into the Hadoop cluster's NameNode	No	`gc.port.cl`
Flume	HDFS Sink	50010	TCP	Communication from Flume into the Hadoop cluster's HDFS DataNode	No	
Flume	HDFS Sink	50020	TCP	Communication from Flume into the Hadoop cluster's HDFS DataNode	No	
Flume	HBase Sink	2181	TCP	Communication from Flume into the Hadoop cluster's Zookeeper	No	
Flume	HBase Sink	60020	TCP	Communication from Flume into the Hadoop cluster's HBase Regionserver	No	
Flume	All Other Sources and Sinks	Variable	Variable	Ports and protocols used by Flume sources and sinks	No	Refer to the f use are specif By default Flu Linux and c:\

HDFS Ports

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

Table 2.3. HDFS Ports

Service	Servers	Default Ports Used
NameNode WebUI	Master Nodes (NameNode and any back-up NameNodes)	50070
		50470
NameNode metadata service		8020/9000
DataNode	All Slave Nodes	50075
		50475
		50010
		50020
Secondary NameNode	Secondary NameNode and any backup Secondanry NameNode	50090

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

Table 2.4. MapReduce Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Config
MapReduce		10020	http	MapReduce JobHistory server address		`mapre`

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Config
MapReduce		19888	http	MapReduce JobHistory webapp address		mapre
MapReduce		13562	http	MapReduce Shuffle Port		mapre

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

Table 2.5. YARN Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
Resource Manager WebUI	Master Nodes (Resource Manager and any back-up Resource Manager node)	8088	http	Web UI for Resource Manager	Yes	`yarn.resourcemanager.webapp`
Resource Manager	Master Nodes (Resource Manager Node)	8050	IPC	For application submissions	Yes (All clients who need to submit the YARN applications including Hive, Hive server, Pig)	Embedded in URI specified by `yarn.resourcemanager.addres`
Resource Manager	Master Nodes (Resource Manager Node)	8025	http	For application submissions	Yes (All clients who need to submit the YARN applications including Hive, Hive server, Pig)	`yarn.resourcemanager.resour tracker.address`
Scheduler	Master Nodes (Resource Manager Node)	8030	http	Scheduler Address	Yes (Typically admins, Dev/ Support teams)	`yarn.resourcemanager.schedu`
Resource Manager	Master Nodes (Resource Manager Node)	8141	http	Scheduler Address	Yes (Typically admins, Dev/ Support teams)	`yarn.resourcemanager.admin.`
NodeManager	Master Nodes (NodeManager) and Slave Nodes	45454	http	NodeManager Address	Yes (Typically admins, Dev/ Support teams)	`yarn.nodemanager.address`
Timeline Server	Master Nodes	10200	http	Timeline Server Address	Yes (Typically admins, Dev/ Support teams)	`yarn.timeline-service.address`
Timeline Server	Master Nodes	8188	http	Timeline Server Webapp Address	Yes (Typically admins, Dev/ Support teams)	`yarn.timeline-service.webapp.address`
Timeline Server	Master Nodes	8190	https	Timeline Server Webapp https Address	Yes (Typically admins, Dev/ Support teams)	`yarn.timeline-service.webapp.https.addres`
Job History Service	Master Nodes	19888	https	Job History Service	Yes (Typically admins, Dev/ Support teams)	`yarn.log.server.url`

Hive Ports

The following table lists the default ports used by the various Hive services. (**Note:** Neither of these services are used in a standard HDP installation.)

Table 2.6. Hive Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
Hive Server	Hive Server machine (Usually a utility machine)	10000		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 Web UI	Hive Server machine (Usually a utility machine)	9999	http	Web UI to explore Hive schemas	Yes	`hive.hwi.listen.port`
Hive Metastore		9933	http		Yes (Clients that run Hive, Pig and potentially M/R jobs that use HCatalog)	`hive.metastore.uris`

HBase Ports

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

Table 2.7. HBase Ports

Service	Servers	Default Ports Used	Proto
HMaster	Master Nodes (HBase Master Node and any back-up HBase Master node)	60000	
HMaster Info Web UI	Master Nodes (HBase master Node and back up HBase Master node if any)	60010	http
Region Server	All Slave Nodes	60020	
Region Server	All Slave Nodes	60030	http
HBase REST Server (optional)	All REST Servers	8080	http
HBase REST Server Web UI (optional)	All REST Servers	8085	http
HBase Thrift Server (optional)	All Thrift Servers	9090	
HBase Thrift Server Web UI (optional)	All Thrift Servers	9095	

Oozie Ports: The following table lists the default ports used by Oozie.

Table 2.8. Oozie Ports

Service	Servers	Default Ports Used	Protocol	Description	N
Oozie	Oozie Server	11000	TCP	The port Oozie server runs.	Y

Service	Servers	Default Ports Used	Protocol	Description	N
Oozie	Oozie Server	11001	TCP	The admin port Oozie server runs.	N
Oozie	Oozie Server	11443	TCP	The port Oozie server runs when using HTTPS.	Y

Sqoop Ports: The following table lists the default ports used by Sqoop.

Table 2.9. Sqoop Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configu
Sqoop	Metastore	16000	TCP	Connection between Sqoop and the metastore	No	`sqoop.`
Sqoop	JDBC Listener	Varies, depends on target database. For example, if moving data from MySQL, TCP port 3306 must be open.	TCP	Outbound port from the Hadoop cluster to the database. Varies depending on Database	No	

Storm Ports

The following table lists the default ports used by Storm.

Table 2.10. Storm Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
Zookeeper Port		2181		Port used by localhost to talk to ZooKeeper.		`storm.zookeeper.port`
DRPC Port		3772				`drpc.port`
DRPC Invocations Port		3773				`drpc.invocations.port`
Nimbus Thrift Port		6627				`nimbus.thrift.port`
Supervisor Slots Ports		6700, 6701, 6702, 7603		Defines the amount of workers that can be run on this machine. Each worker is assigned a port to use for communication.		`supervisor.slots.ports`
Logviewer Port		8000				`logviewer.port`
UI Port		8080				`ui.port`
Ambari Port		8744				`ambari.port`

ZooKeeper Ports

Table 2.11. ZooKeeper Ports

Service	Servers	Default Ports Used	Proto
ZooKeeper Server	All ZooKeeper Nodes	2888	

Service	Servers	Default Ports Used	Proto
ZooKeeper Server	All ZooKeeper Nodes	3888	
ZooKeeper Server	All ZooKeeper Nodes	2181	

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

Table 2.12. MySQL Ports

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

Kerberos Ports: The following table lists the default port used by the designated Kerberos KDC.

Table 2.13. Kerberos Ports

Service	Servers	Default Ports Used	Protocol	Description	Need End User Access?	Configuration Parameters
KDC	Kerberos KDC server	88		Port used by the designated KDC		

3. Controlling HDP Services Manually

In this document:

- Starting HDP Services
- Stopping HDP Services

3.1. Starting HDP Services

Start the Hadoop services in the following order:

- Knox
- ZooKeeper
- HDFS
- YARN
- HBase
- Hive Metastore
- HiveServer2
- WebHCat
- Oozie

Instructions

1. Start Knox. When starting the gateway with the script below, the process runs in the background. The log output is written to `/var/log/knox` and a PID (process ID) is written to `/var/run/knox`. Execute this command on the Knox host machine.

   ```
   cd $GATEWAY_HOME su -l knox -c "bin/gateway.sh start"
   ```

   where *$GATEWAY_HOME* is the is the directory where Knox is installed. For example, `/usr/lib/knox`.

   ### Note

   If Knox has been stopped without using `gateway.sh stop`, you must start the service using `gateway.sh clean`. The clean option removes all log files in `/var/log/knox`.

2. Start ZooKeeper. Execute this command on the ZooKeeper host machine(s):

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

3. Start HDFS

   a. If you are running NameNode HA (High Availability), start the JournalNodes by executing these commands on the JournalNode host machines:

      ```
      su $HDFS_USER
      /usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf start
       journalnode
      ```

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

   b. Execute this command on the NameNode host machine(s):

      ```
      su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
      hadoop/conf start namenode"
      ```

   c. If you are running NameNode HA, start the Zookeeper Failover Controller (ZKFC) by executing the following command on all NameNode machines. The starting sequence of the ZKFCs determines which NameNode will become Active.

      ```
      su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
      hadoop/conf start zkfc"
      ```

   d. If you are not running NameNode HA, execute the following command on the Secondary NameNode host machine. If you are running NameNode HA, the Standby NameNode takes on the role of the Secondary NameNode.

      ```
      su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
      hadoop/conf start secondarynamenode”
      ```

   e. Execute these commands on all DataNodes:

      ```
      su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
      hadoop/conf start datanode"
      ```

4. Start YARN

   a. Execute this command on the ResourceManager host machine(s):

      ```
      su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /
      usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf start
       resourcemanager"
      ```

   b. Execute this command on the History Server host machine:

      ```
      su -l mapred -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec &&
       /usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh --config /etc/
      hadoop/conf start historyserver"
      ```

   c. Execute this command on all NodeManagers:

      ```
      su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /
      usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf start
       nodemanager"
      ```

5. Start HBase

   a. Execute this command on the HBase Master host machine:

```
su -l hbase -c "/usr/lib/hbase/bin/hbase-daemon.sh --config /etc/hbase/
conf start master; sleep 25"
```

b. Execute this command on all RegionServers:

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

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

   ```
   su $HIVE_USER
   nohup hive --service metastore>/var/log/hive/hive.out 2>/var/log/hive/hive.
   log &
   ```

   where, *$HIVE_USER* is the Hive user. For example, `hive`.

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

   ```
   su $HIVE_USER
   nohup /usr/lib/hive/bin/hiveserver2 -hiveconf hive.metastore.uris=" " >>/
   tmp/hiveserver2HD.out 2>> /tmp/hiveserver2HD.log &
   ```

   where, *$HIVE_USER* is the Hive user. For example, `hive`.

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

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

9. Start Oozie. Execute these commands on the Oozie host machine.

   ```
   su $OOZIE_USER
   /usr/lib/oozie/bin/oozie-start.sh
   ```

   where *$OOZIE_USER* is the Oozie user. For example, `oozie`.

3.2. Stopping HDP services

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

- Knox
- Oozie
- WebHCat
- HiveServer2
- Hive Metastore
- HBase
- YARN
- HDFS

- Zookeeper

Instructions

1. Stop Knox. Execute this command on the Knox host machine.

   ```
   cd $GATEWAY_HOME su -l knox -c "bin/gateway.sh stop"
   ```

   where *$GATEWAY_HOME* is the is the directory where Knox is installed. For example, `/usr/lib/knox`.

2. Stop Oozie. Execute these commands on the Oozie host machine.

   ```
   su $OOZIE_USER
   /usr/lib/oozie/bin/oozie-stop.sh
   ```

   where *$OOZIE_USER* is the Oozie user. For example, oozie.

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

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

4. Stop Hive. Execute this command on the Hive Metastore and Hive Server2 host machine.

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

5. Stop HBase

   a. Execute this command on all RegionServers:

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

   b. Execute this command on the HBase Master host machine:

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

6. Stop YARN

   a. Execute this command on all NodeManagers:

      ```
      su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /
      usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop
       nodemanager"
      ```

   b. Execute this command on the History Server host machine:

      ```
      su -l mapred -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec &&
       /usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh --config /etc/
      hadoop/conf stop historyserver"
      ```

   c. Execute this command on the ResourceManager host machine(s):

      ```
      su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /
      usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop
       resourcemanager"
      ```

7. Stop HDFS

a. Execute this command on all DataNodes:

```
su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
hadoop/conf stop datanode"
```

b. If you are not running NameNode HA (High Availability), stop the Secondary NameNode by executing this command on the Secondary NameNode host machine:

```
su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
hadoop/conf stop secondarynamenode”
```

c. Execute this command on the NameNode host machine(s):

```
su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
hadoop/conf stop namenode"
```

d. If you are running NameNode HA, stop the Zookeeper Failover Controllers (ZKFC) by executing this command on the NameNode host machines:

```
su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/
hadoop/conf stop zkfc"
```

e. If you are running NameNode HA, stop the JournalNodes by executing these commands on the JournalNode host machines:

```
su $HDFS_USER
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop
 journalnode
```

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

8. Stop ZooKeeper. Execute this command on the ZooKeeper host machine(s):

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

4. Deploying HDP In Production Data Centers with Firewalls

In this document:

- Deployment strategies for data centers with firewall
- 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. Because 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.

Note

Many of the descriptions in this section assume you are using RHEL/Centos/Oracle Linux. If you are using SLES, please adjust the commands and directories accordingly.

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/ for more information.

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

4.1.2. Mirroring or Proxying

HDP uses yum or zypper to install software, and this software is obtained from the HDP Repositories. If your firewall prevents Internet access, you must mirror or proxy the HDP Repositories in your Data Center.

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 cluster node 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.

- **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.
- **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.
- ### 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.
- **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 cluster node, so that when the client attempts to access the repository during installation, the request goes 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 and II)	Advantages of creating a proxy
• 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, using a mirror for the HDP Repositories can be the best deployment strategy. The HDP Repositories are small and easily mirrored, allowing you secure control over the contents of the Hadoop packages accepted for use in your data center.

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. 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.

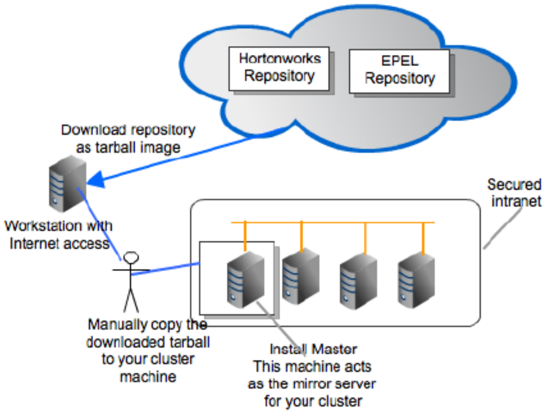
4.3. Detailed Instructions for Creating Mirrors and Proxies

In this section:

- Option I: Mirror server has no access to the Internet
- Option II - Mirror server has temporary or continuous access to the Internet
- 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:


Hortonworks Repository
EPEL Repository
Download repository as tarball image
Workstation with Internet access
Secured intranet
Manually copy the downloaded tarball to your cluster machine
Install Master This machine acts as the mirror server for your cluster

Complete the following instructions to set up a mirror server that has no access to the Internet:

1. Check Your Prerequisites
2. Install the Repos

4.3.1.1. Check Your Prerequisites

Select a mirror server host with the following characteristics:

- This server runs on either CentOS (v5.x, v6.x), RHEL (v5.x, v6.x), Oracle Linux(v5.x, v6.x), SLES 11, or Ubuntu 12, and has several GB of storage available.

- 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 firewall lets all cluster nodes (the servers on which you want to install HDP) to access this server.

4.3.1.2. Install the Repos

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/ Oracle Linux 5.x	`wget http://public-repo-1.hortonworks.com/HDP/centos5/2.x/2.1-latest/HDP-2.1-latest-centos5-rpm.tar.gz`
	`wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.17/repos/centos5/HDP-UTILS-1.1.0.17-centos5.tar.gz`

RHEL/ CentOS/ Oracle Linux 6.x	`wget http://public-repo-1.hortonworks.com/HDP/centos6/2.x/2.1-latest/HDP-2.1-latest-centos6-rpm.tar.gz`
	`wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.17/repos/centos6/HDP-UTILS-1.1.0.17-centos6.tar.gz`

SLES 11	`wget http://public-repo-1.hortonworks.com/HDP/suse11/2.x/2.1-latest/HDP-2.1-latest-suse11-rpm.tar.gz`
	`wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.17/repos/suse11/HDP-UTILS-1.1.0.17-suse11.tar.gz`

Ubuntu 12.04	`wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/HDP-2.1.3.0-ubuntu12-deb.tar.gz`
	`wget http://public-repo-1.hortonworks.com/HDP-UTILS-1.1.0.17/repos/ubuntu12/HDP-UTILS-1.1.0.17-ubuntu12.tar.gz`

2. Create an HTTP server.

   a. On the mirror server, install an HTTP server (such as Apache httpd) using the instructions provided here.

   b. Activate this web server.

   c. 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/Oracle:**

       ```
       mkdir -p /var/www/html/hdp/
       ```

     - **For SLES:**

       ```
       mkdir -p /srv/www/htdocs/rpms
       ```

     - **For Ubuntu:**

       ```
       mkdir -p /var/www/html/hdp/
       ```

   - If you are using a symlink, enable the **followsymlinks** on your web server.

4. Copy the HDP Repository Tarball to the directory created in step 3, and untar it.

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 location: `http://$yourwebserver/hdp/$os/HDP-2.1.3.0/`.

     You should see directory listing for all the HDP components along with the RPMs at: `$os/HDP-2.1.3.0`.

     ### Note

     If you are installing a 2.x.0 release, use: `http://$yourwebserver/hdp/$os/2.x/GA`

     If you are installing a 2.x.x release, use:`http://$yourwebserver/hdp/$os/2.x/updates`

     where

     - `$os` can be `centos5`, `centos6`, `suse11`, or `ubnuntu12`. Use the following options table for `$os` parameter:

Table 4.4. Options for `$os` parameter in repo URL

Operating System	Value
CentOS 5	centos5
RHEL 5	
Oracle Linux 5	
CentOS 6	centos6
RHEL 6	
Oracle Linux 6	
SLES 11	suse11

Operating System	Value
Ubuntu 12	ubuntu12

6. Configure the **yum** clients on all the nodes in your cluster.

   a. Fetch the yum configuration file from your mirror server.

   ```
   http://<$yourwebserver>/hdp/$os/2.x/updates/2.1.3.0/hdp.repo
   ```

   b. Store the `hdp.repo` file to a temporary location.

   c. Edit `hdp.repo` file changing the value of the **baseurl** property to point to your local repositories based on your cluster OS.

   ```
   [HDP-2.x]
   name=Hortonworks Data Platform Version - HDP-2.x
   baseurl=http://$yourwebserver/HDP/$os/2.x
   gpgcheck=1
   gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
   KEY-Jenkins
   enabled=1
   priority=1

   [HDP-UTILS-1.1.0.17]
   name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.17
   baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.17/repos/$os
   gpgcheck=1
   gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
   KEY-Jenkins
   enabled=1
   priority=1

   [HDP-2.1.3.0]
   name=Hortonworks Data Platform HDP-2.1.3.0
   baseurl=http://$yourwebserver/HDP/$os/2.x/updates/2.1.3.0
   gpgcheck=1
   gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
   KEY-Jenkins
   enabled=1
   priority=1
   ```

   where

   - *`$yourwebserver`* is FQDN of your local mirror server.
   - *`$os`* can be `centos5`, `centos6`, `suse11`, or `ubuntu12`. Use the following options table for *`$os`* parameter:

   ### Table 4.5. Options for *$os* parameter in repo URL

Operating System	Value
CentOS 5	centos5
RHEL 5	
Oracle Linux 5	
CentOS 6	centos6

Operating System	Value
RHEL 6	
Oracle Linux 6	
SLES 11	suse11
Ubuntu 12	ubuntu12

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

d. [Conditional]: If you have multiple repositories configured in your environment, deploy the following plugin on all the nodes in your cluster.

i. Install the plugin.

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

```
yum install yum-priorities
```

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

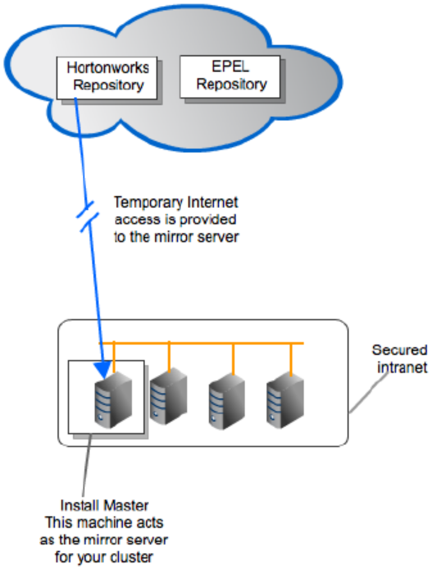
```
yum install yum-plugin-priorities
```

ii. 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 or continuous access to the Internet

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


Hortonworks Repository
EPEL Repository
Temporary Internet access is provided to the mirror server
Secured intranet
Install Master This machine acts as the mirror server for your cluster

Complete the following instructions to set up a mirror server that has temporary access to the Internet:

1. Check Your Prerequisites
2. Install the Repos

4.3.2.1. Check Your Prerequisites

Select a local mirror server host with the following characteristics:

- This server runs on either CentOS/RHEL/Oracle Linux 5.x or 6.x, SLES 11, or Ubuntu 12, and has several GB of storage available.
- The local mirror server and the cluster nodes must have the same OS. If they are not running CentOS or RHEL, the mirror server must not be a member of the Hadoop cluster.

  Note

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

4.3.2.2. Install the Repos

- Temporarily reconfigure your firewall to allow Internet access from your mirror server host.
- 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 4.6. Deploying HDP - Option II

Cluster OS	HDP Repository Tarballs
RHEL/CentOS/Oracle Linux 5.x	`wget http://public-repo-1.hortonworks.com/HDP/centos5/2.x/2.1-latest/hdp.repo -O /etc/yum.repos.d/hdp.repo`
RHEL/CentOS/Oracle Linux 6.x	`wget http://public-repo-1.hortonworks.com/HDP/centos6/2.x/2.1-latest/hdp.repo -O /etc/yum.repos.d/hdp.repo`
SLES 11	`wget http://public-repo-1.hortonworks.com/HDP/suse11/2.x/2.1-latest/hdp.repo -O /etc/zypp/repos.d/hdp.repo`

Cluster OS	HDP Repository Tarballs
Ubuntu 12.04	`wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.1.3.0/hdp.list -O /etc/apt/sources.list.d/hdp.list`

- Create an HTTP server.

  1. On the mirror server, install an HTTP server (such as Apache `httpd`) using the instructions provided http://httpd.apache.org/download.cgi

  2. Activate this web server.

  3. 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.

  4. 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
     ```

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

  - For example, from a shell window, type:

    - **For RHEL/CentOS/Oracle:**

      ```
      mkdir -p /var/www/html/hdp/
      ```

    - **For SLES:**

      ```
      mkdir -p /srv/www/htdocs/rpms
      ```

    - **For Ubuntu:**

      ```
      mkdir -p /var/www/html/hdp/
      ```

  - If you are using a symlink, enable the `followsymlinks` on your web server.

- Copy the contents of entire HDP repository for your desired OS from the remote yum server to your local mirror server.

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

    - **For RHEL/CentOS/Oracle/Ubuntu:**

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

    - **For SLES:**

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

Then for all hosts, type:

- HDP Repository

  ```
  reposync -r HDP
  reposync -r HDP-2.1.3.0
  reposync -r HDP-UTILS-1.1.0.17
  ```

  You should see both an `HDP-2.1.3.0` directory and an `HDP-UTILS-1.1.0.17` directory, each with several subdirectories.

- Generate appropriate metadata.

  This step defines each directory as a yum repository. From a shell window, type:

  - For RHEL/CentOS/Oracle:

    - HDP Repository:

      ```
      createrepo /var/www/html/hdp/HDP-2.1.3.0
      createrepo /var/www/html/hdp/HDP-UTILS-1.1.0.17
      ```

  - For SLES:

    - HDP Repository:

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

  You should see a new folder called `repodata` inside both HDP directories.

- 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 location:

    - HDP: **http://*$yourwebserver*/hdp/HDP-2.1.3.0/**

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

- At this point, you can disable external Internet access for the mirror server, so that the mirror server is again entirely within your data center firewall.

- Depending on your cluster OS, configure the **yum** clients on all the nodes in your cluster

  1. Edit the repo files, changing the value of the baseurl property to the local mirror URL.

     - Edit the `/etc/yum.repos.d/hdp.repo` file, changing the value of the `baseurl` property to point to your local repositories based on your cluster OS.

       ```
       [HDP-2.x]
       name=Hortonworks Data Platform Version - HDP-2.x
       baseurl=http://$yourwebserver/HDP/$os/2.x
       gpgcheck=1
       gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
       KEY-Jenkins
       ```

```
enabled=1
priority=1

[HDP-UTILS-1.1.0.17]
name=Hortonworks Data Platform Utils Version - HDP-UTILS-1.1.0.17
baseurl=http://$yourwebserver/HDP-UTILS-1.1.0.17/repos/$os
gpgcheck=1
gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
KEY-Jenkins
enabled=1
priority=1

[HDP-2.1.3.0]
name=Hortonworks Data Platform HDP-2.1.3.0
baseurl=http://$yourwebserver/HDP/$os/2.x/updates/2.1.3.0
gpgcheck=1
gpgkey=http://public-repo-1.hortonworks.com/HDP/$os/RPM-GPG-KEY/RPM-GPG-
KEY-Jenkins
enabled=1
priority=1
```

where

- *$yourwebserver* is FQDN of your local mirror server.
- *$os* can be `centos5`, `centos6`, or `suse11`. Use the following options table for *$os* parameter:

Table 4.7. Options for *$os* parameter in repo URL

Operating System	Value
CentOS 5	centos5
RHEL 5	
Oracle Linux 5	
CentOS 6	centos6
RHEL 6	
Oracle Linux 6	
SLES 11	suse11
Ubuntu 12	ubuntu12

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

- RHEL/CentOS/Oracle Linux:

  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/suse11/2.x/updates/2.1.3.0/hdp.repo**

- For Ubuntu:

On every node, invoke the following command:

- HDP Repository: **sudo add-apt-repository 'deb http://$yourwebserver/hdp/HDP/ubuntu12/2.x/hdp.list'**
- Optional - Ambari Repository: **sudo add-apt-repository 'deb http://$yourwebserver/hdp/ambari/ubuntu12/1.x/updates/1.5.1/ambari.list'**
- If using Ambari, verify the configuration by deploying Ambari server on one of the cluster nodes. **yum install ambari-server**

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

  1. Install the plugin.

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

       ```
       yum install yum-priorities
       ```

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

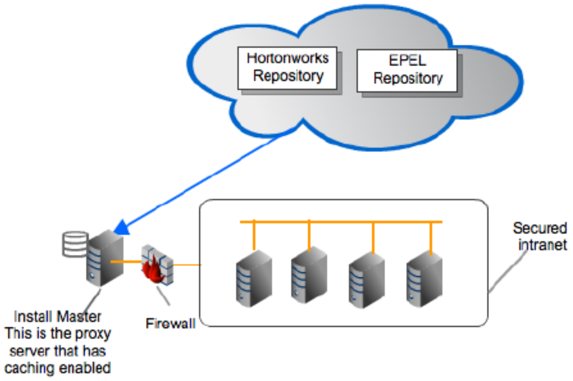
       ```
       yum install yum-plugin-priorities
       ```

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

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

4.3.3. Set up a trusted proxy server

A trusted proxy server is set up in the following illustration:



Complete the following instructions to set up a trusted proxy server:

1. Check Your Prerequisites
2. Install the Repos

4.3.3.1. Check Your Prerequisites

Select a mirror server host with the following characteristics:

- This server runs on either CentOS/RHEL/Oracle Linux (5.x or 6.x), SLES 11, or Ubuntu 12, 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).

4.3.3.2. Install the Repos

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** clients on all the nodes in your cluster.

   - **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/
   ```

5. Wire Encryption in Hadoop

This section provides an overview on encryption over-the-wire in Hadoop. Data can be moved in and out of Hadoop over RPC, HTTP, Data Transfer Protocol, and JDBC. Network traffic over each of these protocols can be encrypted to provide privacy for data movement.

1. Enable RPC Encryption for the Hadoop Cluster
2. Enable Data Transfer Protocol
3. Enable SSL for HTTP Connections
4. Enable SSL with HiveServer2 JDBC Connections

5.1. Enable RPC Encryption for the Hadoop Cluster

The most common way for a client to interact with a Hadoop cluster is through RPC. A client connects to a NameNode over RPC protocol to read or write a file. RPC connections in Hadoop use the Java Simple Authentication and Security Layer (SASL) which supports encryption. When the `hadoop.rpc.protection` property is set to privacy, the data over RPC is encrypted with symmetric keys.

Note

RPC encryption covers not only the channel between a client and a Hadoop cluster but also the inter-cluster communication among Hadoop services.

Enable Encrypted RPC by setting the follwing properties in core-site.xml.

```
hadoop.rpc.protection=privacy
```

(Also supported are the ‘authentication’ and ‘integrity’ settings.)

5.2. Enable Data Transfer Protocol

The NameNode gives the client the address of the first DataNode to read or write the block. The actual data transfer between the client and the DataNode is over Hadoop's Data Transfer Protocol. To encrypt this protocol you must set `dfs.encryt.data.transfer=true` on the NameNode and all DataNodes. The actual algorithm used for encryption can be customized with `dfs.encrypt.data.transfer.algorithm` set to either "3des" or "rc4". If nothing is set, then the default on the system is used (usually 3DES.) While 3DES is more cryptographically secure, RC4 is substantially faster.

Enable Encrypted DTP by setting the following properties in `hdfs-site.xml`:

```
 dfs.encrypt.data.transfer=true
dfs.encrypt.data.transfer.algorithm=3des
```

(‘rc4’is also supported.)

Note

Secondary Namenode is not supported with the HTTPS port. It can only be accessed by “http://<SNN>:50090”. WebHDFS, hsftp, and shortcircuitread are not supported with SSL enabled.

5.3. Enable SSL for HTTP Connections

Encryption over the HTTP can be implemented with support for SSL across your Hadoop cluster.

5.3.1. Set up WebHDFS/YARN with SSL (HTTPS)

This section explains how to set up HTTPS encryption for the Web interfaces.

5.3.1.1. Install an SSL Certificate

You can use either a certificate from a Certificate Authority or a Self-Signed Certificate. Using a self-signed certificate requires some additional configuration on each host. Follow the instructions in the appropriate section to install a certificate.

5.3.1.1.1. Use a Self-Signed Certificate

To set up SSL for Hadoop HDFS operations:

1. Create HTTPS certificates and keystore/truststore files.

   a. For each host in the cluster, create a directory for storing the keystore and truststore. For example, SERVER_KEY_LOCATION. Also create a directory to store public certificate, for example, CLIENT_KEY_LOCATION.

   ```
   mkdir -p $SERVER_KEY_LOCATION ; mkdir -p $CLIENT_KEY_LOCATION
   ```

   For example:

   ```
   ssh host1.hwx.com “mkdir -p /etc/security/serverKeys ; mkdir -p /etc/
   security/clientKeys ; ”
   ```

   b. For each host, create a keystore file.

   ```
   cd $SERVER_KEY_LOCATION ; keytool -genkey -alias $hostname -keyalg RSA
    -keysize 1024 -dname \"CN=$hostname,OU=hw,O=hw,L=paloalto,ST=ca,C=us\
   " -keypass $SERVER_KEYPASS_PASSWORD -keystore $KEYSTORE_FILE -storepass
    $SERVER_STOREPASS_PASSWORD\”
   ```

   c. For each host, export the certificate public key to a certificate file.

   ```
   cd $SERVER_KEY_LOCATION ; keytool -export -alias $hostname -
   keystore $KEYSTORE_FILE -rfc -file $CERTIFICATE_NAME -storepass
    $SERVER_STOREPASS_PASSWORD\”
   ```

   d. For each host, import the certificate into truststore file.

   ```
   cd $SERVER_KEY_LOCATION ; keytool -import -noprompt -alias $hostname
    -file $CERTIFICATE_NAME -keystore $TRUSTSTORE_FILE -storepass
    $SERVER_TRUSTSTORE_PASSWORD
   ```

e. Create a single truststore file containing the public key from all certificates. Login to host1 and import the truststore file for host1.

```
keytool -import -noprompt -alias $host -file $CERTIFICATE_NAME -keystore
 $ALL_JKS -storepass $CLIENT_TRUSTSTORE_PASSWORD
```

f. Copy $ALL_JKS from host1 to other hosts, and repeat the above command. For example, for a 2-node cluster with host1 and host2:

   i. Create $ALL_JKS on host1.

   ```
   keytool -import -noprompt -alias $host -file $CERTIFICATE_NAME -
   keystore $ALL_JKS -storepass $CLIENT_TRUSTSTORE_PASSWORD
   ```

   ii. Copy over $ALL_JKS from host1 to host2. $ALL_JKS already has the certificate entry of host1.

   iii. Import certificate entry of host2 to $ALL_JKS using same command as before:

   ```
   keytool -import -noprompt -alias $host -file $CERTIFICATE_NAME -
   keystore $ALL_JKS -storepass $CLIENT_TRUSTSTORE_PASSWORD
   ```

   iv. Copy over the updated $ALL_JKS from host2 to host1.

   Note

   Repeat these steps each time for each node in the cluster. When you are finished, the $ALL_JKS file on host1 will have the certificates of all nodes.

   v. Copy over the $ALL_JKS file from host1 to all the nodes.

g. Validate the common truststore file on all hosts.

```
keytool -list -v -keystore $ALL_JKS -storepass
 $CLIENT_TRUSTSTORE_PASSWORD
```

h. Set permissions and ownership on the keys:

```
chgrp -R $YARN_USER:hadoop $SERVER_KEY_LOCATION
chgrp -R $YARN_USER:hadoop $CLIENT_KEY_LOCATION
chown 755 $SERVER_KEY_LOCATION
chown 755 $CLIENT_KEY_LOCATION
chown 440 $KEYSTORE_FILE
chown 440 $TRUSTSTORE_FILE
chown 440 $CERTIFICATE_NAME
chown 444 $ALL_JKS
```

Note

The complete path of the `$SEVER_KEY_LOCATION` and the `CLIENT_KEY_LOCATION` from the root directory /etc must be owned by the $YARN_USER user and the hadoop group.

5.3.1.1.2. Use a CA Signed Certificate

1. Run the following command to create a self-signing rootCA and import the rootCA into client truststore:

   ```
   openssl genrsa -out $clusterCA.key 2048
   openssl req -x509 -new -key $clusterCA.key -days 300 -out $clusterCA.pem
   keytool -importcert -alias $clusterCA -file $clusterCA.pem -keystore
    $clustertruststore -storepass $clustertruststorekey
   ```

   Note

   Ensure that the ssl-client.xml on every host configure to use this '$clustertrust' store.

2. On each host, run the following command to create a certifcate and a keystore for each server:

   ```
   keytool -genkeypair -alias `hostname -s` -keyalg RSA -keysize 1024 -dname
    "CN=`hostname -f`,OU=foo,O=corp" -keypass $hostkey -keystore $hostkeystore
    -storepass $hoststorekey -validity 300
   ```

3. On each host, run the following command to export a certreq file from the host's keystore:

   ```
   keytool -keystore keystore -alias `hostname -s` -certreq -file $host.cert -
   storepass $hoststorekey -keypass $hostkey
   ```

4. On each host, sign certreq file with the rootCA:

   ```
   openssl x509 -req -CA $clusterCA.pem -CAkey $clusterCA.key -in $host.cert -
   out $host.signed -days 300 -CAcreateserial
   ```

5. On each host, import both rootCA and the signed cert back in:

   ```
   keytool -keystore $hostkeystore -storepass $hoststorekey -alias $clusterCA -
   import -file cluseter1CA.pem
   keytool -keystore $hostkeystore -storepass $hoststorekey -alias `hostname -
   s` -import -file $host.signed -keypass $hostkey
   ```

5.3.1.2. Set Hadoop Properties to Enable HTTPS

To enable WebHDFS to listen for HTTP over SSL, configure SSL on the NameNode and all DataNodes by setting `dfs.https.enable=true` in the `hdfs-site.xml` file.

You can set up SSL in the following modes:

- One-way SSL: Authenticates the server only. This mode requires the keystore on the NameNode and each DataNode only as specified in the table below. The parameters are set in the `ssl-server.xml` file on the NameNode and each of the DataNodes.
- Mutual authentication (2WAY SSL): Requires authentication of both the server and the client. To use mutual SSL, you must also set `dfs.client.https.need-auth=true` in the `hdfs-site.xml` file on the NameNode and each DataNode. 2WAY SSL can cause performance delays and is difficult to set up and maintain.

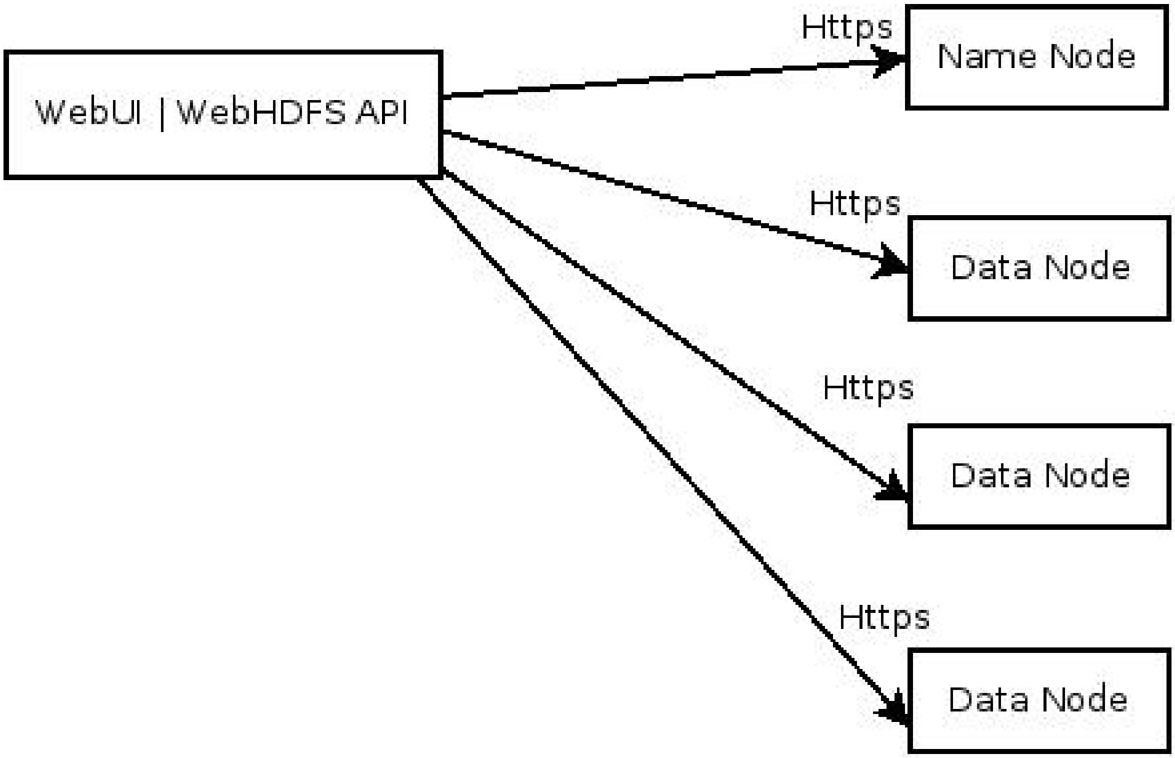
The truststore configuration is only needed when using a self-signed certificate or a certificate that is not in the JVM's truststore.

The following configuration properties need to be specified in ssl-server.xml and ssl-client.xml.

Table 5.1. Configuration Properties in ssl-server.xml

Property	Default Value	Description
ssl.server.keystore.type	JKS	The type of the keystore, JKS = Java Keystore, the de-facto standard in Java
ssl.server.keystore.location	None	The location of the keystore file
ssl.server.keystore.password	None	The password to open the keystore file
ssl.server truststore.type	JKS	The type of the trust store
ssl.server.truststore.location	None	The location of the truststore file
ssl server.truststore.password	None	The password to open the trustsore

The following diagram shows an HTTP or REST client's interaction with the NameNode and the DataNodes over HTTPS.



Enable HTTPS by setting the following properties.

1. Set the following properties in `core-site.xml`.

```
hadoop.ssl.require.client.cert=false
hadoop.ssl.hostname.verifier=DEFAULT
hadoop.ssl.keystores.factory.class=org.apache.hadoop.security.ssl.
FileBasedKeyStoresFactory
hadoop.ssl.server.conf=ssl-server.xml
hadoop.ssl.client.conf=ssl-client.xml
```

2. Set the following properties in `ssl-server.xml`.

```
ssl.server.truststore.location=/etc/security/serverKeys/truststore.jks
ssl.server.truststore.password=serverTrustStorePassword
ssl.server.truststore.type=jks
ssl.server.keystore.location=/etc/security/serverKeys/keystore.jks
ssl.server.keystore.password=serverStorePassPassword
ssl.server.keystore.type=jks
ssl.server.keystore.keypassword=serverKeyPassPassword
```

3. Set the following properties in `ssl-client.xml`.

   ```
   ssl.client.truststore.location=/etc/security/clientKeys/all.jks
   ssl.client.truststore.password=clientTrustStorePassword
   ssl.client.truststore.type=jks
   ```

4. Set the following properties in `hdfs-site.xml`.

   ```
   dfs.http.policy=$Policy
   dfs.datanode.https.address=$DataNode-host:50475
   dfs.namenode.https-address=$NAMENODE-host:50470
   ```

   where $*Policy* is either:

   - HTTP_ONLY: Service is provided only on HTTP

   - HTTPS_ONLY: Service is provided only on HTTPS

   - HTTP_AND_HTTPS: Service is provided both on HTTP and HTTPS

5. Set the following properties in `mapred-site.xml`:

   ```
   mapreduce.jobhistory.http.policy=HTTPS_ONLY
   mapreduce.jobhistory.webapp.https.address=<JHS>:<JHS_HTTPS_PORT>
   ```

6. Set the following properties in `yarn-site.xml`:

   ```
   yarn.http.policy=HTTPS_ONLY
   yarn.log.server.url=https://<JHS>:<JHS_HTTPS_PORT>/jobhistory/logs
   yarn.resourcemanager.webapp.https.address=<RM>:<RM_HTTPS_PORT>
   yarn.nodemanager.webapp.https.address=0.0.0.0:<NM_HTTPS_PORT>
   ```

5.3.2. Configure Encryption during a Shuffle

Data securely loaded into HDFS is processed by Mappers and Reducers to derive meaningful business intelligence. Hadoop code moves data between Mappers and Reducers over the HTTP protocol in a step called the shuffle. In SSL parlance, the Reducer is the SSL client that initiates the connection to the Mapper to ask for data. Enabling HTTPS for encrypting shuffle traffic involves the following steps.

- Enable Encrypted Shuffle by setting the follwing properties in `mapred-site.xml`:

  ```

  <property>
      <name>hadoop.ssl.enabled</name>
      <value>true</value>
  </property>
  <property>
      <name>hadoop.ssl.require.client.cert</name>
      <value>false</value>
  ```

```
    <final>true</final>
</property>
<property>
    <name>hadoop.ssl.hostname.verifier</name>
    <value>DEFAULT</value>
    <final>true</final>
</property>
<property>
    <name>hadoop.ssl.keystores.factory.class</name>
    <value>org.apache.hadoop.security.ssl.FileBasedKeyStoresFactory</value>
    <final>true</final>
</property>
<property>
    <name>hadoop.ssl.server.conf</name>
    <value>ssl-server.xml</value>
    <final>true</final>
</property>
<property>
    <name>hadoop.ssl.client.conf</name>
    <value>ssl-client.xml</value>
    <final>true</final>
</property>
```

(The default buffer size is 65536.)

5.3.3. Set up Ozzie with SSL (HTTPS)

The default SSL configuration makes all Oozie URLs use HTTPS except for the JobTracker callback URLs. This simplifies the configuration because no changes are required outside of Oozie. Oozie inherently does not trust the callbacks, they are used as hints.

Note

The related environment variables are explained at Environment Setup .

5.3.3.1. Install the SSL Certificate

You can use either a certificate from a Certificate Authority or a Self-Signed Certificate. Using a self-signed certificate requires some additional configuration on each Oozie client machine.

Follow the instructions in the appropriate section to install a certificate.

5.3.3.1.1. Create a Self-Signed Certificate

There are many ways to create a Self-Signed Certificate, this is just one way. We will be using the keytool program, which is included with your JRE. If its not on your path, you should be able to find it in *$JAVA_HOME*/bin.

1. As the Oozie user, run the following command:

   ```
   keytool -genkey -alias tomcat -keyalg RSA
   ```

2. An interactive prompt displays, answer the questions as follows:

   - For **What is your first and last name?** (i.e. "CN") enter the hostname of the machine where the Oozie Server is running.

- For the **keystore password** and **key password**, enter the same password; by default, the Oozie keystore password is set to *`password`*.

A keystore file is created named `.keystore` and is located in the Oozie user's home directory.

3. Change the OOZIE_HTTPS_KEYSTORE_PASS environment variable to match the *keystore password* of the self-signed certificate.

4. As the Oozie user, run the following command to export a certificate file from the keystore file:

   ```
   keytool -exportcert -alias tomcat -file path/to/where/I/want/my/certificate.cert
   ```

5.3.3.1.2. Use a Certificate from a Certificate Authority

Consult a Certificate Authority to obtain a certicate. Use the certificate from the CA to set up Oozie with HTTPS.

As the Oozie user, run the following command to create a keystore file from your certificate:

```
keytool -import -alias tomcat -file path/to/certificate.cert
```

A keystore file is created named `.keystore` and is located in the Oozie user's home directory.

5.3.3.2. Set up Oozie Server Secure Mode

This describes how to configure Oozie to use HTTPS instead of HTTP.

1. If Oozie server is running, stop Oozie.

2. As the Oozie user, run the following command to configure Oozie to use HTTPS:

   ```
   oozie-setup.sh prepare-war -secure
   ```

3. Start the Oozie server.

Note

To revert back to HTTP, as the Oozie user run the following command:

```
oozie-setup.sh prepare-war
```

5.3.3.3. Connect to Oozie Server using SSL (HTTPS)

On every Oozie client system, follow the instructions for the type of certificate used in your environment.

5.3.3.3.1. Use a Self-signed Certificate from Oozie Clients

When using a self-signed certificate, you must first install the certificate before the Oozie client can connect to the server.

1. Install the certificate in the keychain:

    a. Copy or download the .cert file onto the client machine.

    b. Run the following command (as root) to import the certificate into the JRE's keystore:

    ```
    sudo keytool -import -alias tomcat -file path/to/certificate.cert -
    keystore $JRE_cacerts
    ```

    Where *`$JRE_cacerts`* is the path to the JRE's certs file. It's location may differ depending on the Operating System, but its typically called cacerts and located at *$JAVA_HOME*/lib/security/cacerts. It can be under a different directory in *$JAVA_HOME*. The default password is `changeit`.

    Java programs, including the Oozie client, can now connect to the Oozie Server using the self-signed certificate.

2. In the connection strings change HTTP to HTTPS, for example, replace `http://oozie.server.hostname:11000/oozie` with `https://oozie.server.hostname:11443/oozie`.

    Java does not automatically redirect HTTP addresses to HTTPS.

5.3.3.3.2. Use a CA Certificate from Oozie Clients

In the connection strings change HTTP to HTTPS, for example, replace `http://oozie.server.hostname:11000/oozie` with `https://oozie.server.hostname:11443/oozie`.

Java does not automatically redirect HTTP addresses to HTTPS.

5.3.3.4. Connect to Oozie Server using a Browser

Use `https://oozie.server.hostname:11443/oozie` though most browsers should automatically redirect you if you use `http://oozie.server.hostname:11000/oozie`.

When using a Self-Signed Certificate, your browser warns you that it can't verify the certificate. Add the certificate as an exception.

5.3.3.5. Configure Oozie HCatalogJob Properties

Integrate Oozie Hcatalog by adding following property to oozie-hcatalog job.properties. For example if you are using Ambari, set the properties as:

```
hadoop.rpc.protection=privacy
```

Note

This property is in addition to any properties you must set for secure clusters.

5.3.4. Set up HBase REST API with SSL

Perform the following task to enable SSL with the HBase REST API.

1. Execute the following statement from the command line of the HBase Master server to create a keystore for HBase:

   ```
   keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks
   ```

2. Add the following properties to the `hbase-site.xml` configuration file on each node in your HBase cluster:

   ```
   <property>
       <name>hbase.rest.ssl.enabled</name>
       <value>true</value>
   </property>

   <property>
       <name>hbase.rest.ssl.keystore.store</name>
       <value>/path/to/keystore</value>
   </property>

   <property>
       <name>hbase.rest.ssl.keystore.password</name>
       <value>keystore password</value>
   </property>

   <property>
       <name>hbase.rest.ssl.keystore.keypassword</name>
       <value>key password</value>
   </property>
   ```

3. Restart all HBase nodes in the cluster.

Note

When using a self-signed certificate, administrators must manually add the certificate to the JVM truststore on all HBase clients.

5.3.5. Set up WebHBase with SSL

1. On the HBase Master, create keystore for Hbase:

   ```
   keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks
   ```

2. Add the following properties to `hbase-site.xml`:

   ```
   <property>
      <name>hadoop.ssl.enabled</name>
      <value>true</value>
   </property>

   <property>
      <name>hbase.rest.ssl.enabled</name>
      <value>true</value>
   </property>

   <property>
      <name>hbase.rest.ssl.keystore.store</name>
      <value>/path/to/hbase.jks</value>
   </property>
   ```

```
<property>
   <name>hbase.rest.ssl.keystore.password</name>
   <value> keystore password</value>
</property>

<property>
   <name>hbase.rest.ssl.keystore.keypassword</name>
   <value> key password</value>
</property>
```

3. Restart HBase.

   When using a self-signed certificate, you must add the certificate to JMV truststore on the HBase clients.

5.3.6. Set up HiveServer2 with SSL

When using HiveServer2 without Kerberos authentication, you can set up HTTP and JDBC to use an SSL certificate to secure communications.

Perform the followin steps on the HiveServer2:

1. Run the following command to create a keystore for hiveserver2::

   ```
   keytool -genkey -alias hbase -keyalg RSA -keysize 1024 -keystore hbase.jks
   ```

2. Edit the `hive-site.xml`, set the following properties to enable SSL:

   ```
   <property>
     <name>hive.server2.enable.SSL</name>
     <value>true</value>
     <description></description>
   </property>

   <property>
     <name>hive.server2.keystore.path</name>
     <value>$keystore-file-path</value>
     <description></description>
   </property>

   <property>
     <name>hive.server2.keystore.password</name>
     <value>$keystore-file-password</value>
     <description></description>
   </property>
   ```

3. On the client-side, specify SSL settings for Beeline or JDBC client as follows:

   ```
   jdbc:hive2://$host:$port/$database;ssl=true;sslTrustStore=$path-to-
   truststore;sslTrustStorePassword=$password
   ```

5.4. Enable SSL for HiveServer2 JDBC Connections

HiveServer2 implemented encryption with the Java SASL protocol's quality of protection (QOP) setting that allows data moving between a HiveServer2 over JDBC and a JDBC client to be encrypted.

From the JDBC client specify `sasl.sop` as part of the JDBC-Hive connection string, for example `jdbc:hive://hostname/dbname;sasl.qop=auth-int`.

Tip

See HIVE-4911 for more details on this enhancement.

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	PostgreSQL 9.x
RHEL/Centos/Oracle Linux 5.x RHEL/CentOS/Oracle Linux 6.x SLES 11 Ubuntu 12	Hive / HCatalog	Supported. For instructions on configuring this database for Hive metastore, see Instructions for Manual Install.	Supported. For instructions on c metastore, see Instructions for M
	Oozie	Supported. For instructions on configuring this database for Oozie metastore, see Instructions for Manual Install.	Supported. For instructions on c metastore, see Instructions for M
	Hue [a]	Supported. For instructions on configuring this database for Hue, see Instructions for Manual Install.	Supported. For instructions on c Instructions for Manual Install .
	Ambari [b]	Default. For more information, see Database Requirements.	Supported. For more informatio

[a] Hue does not currently support Ubuntu 12.

[b] Ambari does not currently support Ubuntu 12.