Hortonworks Data Platform

Installing HDP Manually

(Jun 12, 2013)

docs.hortonworks.com

Hortonworks Data Platform : Installing HDP Manually

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.

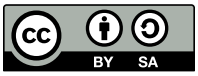

Table of Contents

List of Tables

1. Getting Ready to Install

This section describes the information and materials you need to get ready to install the Hortonworks Data Platform (HDP) manually. In general, the following instructions cover non-secure installations. For the additional information and steps needed to add security (Kerberos) to your installation, please see Setting Up Security for Manual Installs.

In this section:

1.1. Understand the Basics

The Hortonworks Data Platform consists of three layers.

- **Core Hadoop**: The basic components of Apache Hadoop.
  - **Hadoop Distributed File System (HDFS)**: A special purpose file system that is designed to work with the MapReduce engine. It provides high-throughput access to data in a highly distributed environment.
  - **MapReduce**: A framework for performing high volume distributed data processing using the MapReduce programming paradigm.
- **Essential Hadoop:** A set of Apache components designed to ease working with Core Hadoop.
  - **Apache Pig**: A platform for creating higher level data flow programs that can be compiled into sequences of MapReduce programs, using Pig Latin, the platform's native language.
  - **Apache Hive**: A tool for creating higher level SQL queries using HiveQL, the tool's native language, that can be compiled into sequences of MapReduce programs.
  - **Apache HCatalog**: A metadata abstraction layer that insulates users and scripts from how and where data is physically stored.

- **WebHCat (Templeton)**: A component that provides a set of REST APIs for HCatalog and related Hadoop components.
- **Apache HBase**: A distributed, column-oriented database that provides the ability to access and manipulate data randomly in the context of the large blocks that make up HDFS.
- **Apache ZooKeeper**:A centralized tool for providing services to highly distributed systems. ZooKeeper is necessary for HBase installations.

- **Supporting Components**: A set of components that allow you to monitor your Hadoop installation and to connect Hadoop with your larger compute environment.
  - **Apache Oozie**:A server based workflow engine optimized for running workflows that execute Hadoop jobs.
  - **Apache Sqoop**: A component that provides a mechanism for moving data between HDFS and external structured datastores. Can be integrated with Oozie workflows.
  - **Apache Flume**: A log aggregator. This component must be installed manually. See Installing and Configuring Flume NG for more information.
  - **Apache Mahout**: A scalable machine learning library that implements several different approaches to machine learning. This component must be installed manually on an appropriate host, using yum for RHEL or CentOS or zypper for SLES. No configuration is needed.
  - **Ganglia**: An Open Source tool for monitoring high-performance computing systems.
  - **Nagios**: An Open Source tool for monitoring systems, services, and networks.

You must always install Core Hadoop, but you can select the components from the other layers based on your needs.

For more information on the structure of the HDP, see Understanding Hadoop Ecosystem.

1.2. Meet Minimum System Requirements

To run the Hortonworks Data Platform, your system must meet minimum requirements.

- Hardware Recommendations
- Operating System Requirements
- Software Requirements
- Configure the Remote Repository
- Database Requirements
- JDK Recommendations
- Virtualization and Cloud Platforms

1.2.1. Hardware Recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. You can see sample setups here: Hardware Recommendations for Apache Hadoop.

1.2.2. Operating Systems Requirements

The following operating systems are supported:

- 64-bit Red Hat Enterprise Linux (RHEL) 5 or 6
- 64-bit CentOS 5 or 6
- 64-bit SUSE Linux Enterprise Server (SLES) 11, SP1

1.2.3. Software Requirements

On each of your hosts:

- yum [for RHEL or CentOS]
- zypper [for SLES]
- rpm
- scp [for multiple node installs]
- curl
- wget
- unzip
- tar
- pdsh [for multiple node installs over many hosts]

1.2.4. Configure the Remote Repository

The standard HDP install fetches the software from a remote yum repository over the Internet. To use this option, you must set up access to the remote repository and have an available Internet connection for each of your hosts.

Note

If your cluster does not have access to the Internet, or you are creating a large cluster and you want to conserve bandwidth, you can instead provide a local copy of the HDP repository that your hosts can access. For more information, see Deployment Strategies for Data Centers with Firewalls., a separate document in this set.

1. For each node in your cluster, download the repo configuration file `hdp.repo` and `ambari.repo`. From a terminal window, type:

   - For RHEL and CentOS 5

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/1.x/GA/1.3.0.0/
     hdp.repo  -O /etc/yum.repos.d/hdp.repo
      wget -nv http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/
     1.2.4.9/ambari.repo -O /etc/yum.repos.d/ambari.repo
     ```

   - For RHEL and CentOS 6

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/1.x/GA/1.3.0.0/
     hdp.repo -O /etc/yum.repos.d/hdp.repo
      wget -nv http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/
     1.2.4.9/ambari.repo -O /etc/yum.repos.d/ambari.repo
     ```

   - For SLES

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/suse11/1.x/GA/1.3.0.0/
     hdp.repo -O /etc/zypp/repos.d/hdp.repo
      wget -nv http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/
     1.2.4.9/ambari.repo -O /etc/zypp/repos.d/ambari.repo
     ```

2. Confirm the HDP repository is configured by checking the repo list.

   - For RHEL/CentOS

     ```
     yum repolist
     ```

   - For SLES

     ```
     zypper repos
     ```

You should see something like this. Ensure you have HDP-1.3.0.0, HDP-UTILS-1.1.0.15, and AMBARI-1.2.3.7:

```
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
* base: mirrors.cat.pdx.edu
* extras: linux.mirrors.es.net
* updates: mirrors.usc.edu
repo id                                    repo name
                                                                    status
AMBARI-1.2.4.9                             Ambari 1.2.4.9
                                  enabled:      6
HDP-1.3.0.0                                Hortonworks Data Platform Version -
 HDP-1.3.0                                 enabled:     53
HDP-UTILS-1.1.0.15                         Hortonworks Data Platform Utils
 Version - HDP-UTILS-1.1.0.15              enabled:     51
```

1.2.5. Database Requirements

- To use external database for Hive or Oozie metastore, ensure that a MySQL or Oracle or PostgreSQL database is deployed and available.

  (By default, Hive and Oozie use Derby database for the metastore.)

- For instructions on deploying and/or configuring MySQL database instance, see here [5].
- For instructions on configuring an existing Oracle database instance, see here [6].

Note

To deploy a new Oracle instance, consult your database administrator.

- For instructions on deploying and/or configuring an existing PostgreSQL database instance, see here [6].

- Ensure that your database administrator creates the following databases and users:
  - If deploying Hive:
    1. hive_dbname: Required if using MySQL database for Hive Metastore.
    2. hive_dbuser
    3. hive_dbpasswd
  - If deploying Oozie:
    1. oozie_dbname: Required if using MySQL database for Oozie Metastore.
    2. oozie_dbuser
    3. oozie_dbpasswd

Instructions to setup MySQL database

1. Connect to the host machine where you plan to deploy MySQL instance and from a terminal window, type:
   - For RHEL and CentOS:

     ```
     yum install mysql
     ```

   - For SLES:

     ```
     zypper install mysql
     ```

2. Start the instance.
   - For RHEL and CentOS:

     ```
     /etc/init.d/mysqld start
     ```

   - For SLES:

     ```
     /etc/init.d/mysql start
     ```

3. Set the `root` user password and remove unnecessary information from `log` and STDOUT.

```
mysqladmin -u root password '$password'
```

```
mysqladmin -u root 2>&1 >/dev/null
```

4. Manually create users for MySQL.

   - As `root`, use mysql (or other client tool) to create the "dbuser" and grant it adequate privileges.

     (For access to Hive metastore, create `hive_dbuser` and for access to Oozie metastore, create `oozie_dbuser`.

     ```
     CREATE USER 'dbusername'@'%' IDENTIFIED BY 'dbuserpassword';
     GRANT ALL PRIVILEGES ON *.* TO 'dbusername'@'%';
     flush privileges;
     ```

   - See if you can connect to the database as that user. You are prompted to enter the `dbuserpassword` password above.

     ```
     mysql -u $dbusername -p
     ```

Instructions to configure Oracle database

- Ensure that the following SQL script is run against your Hive schema:

```
$master-install-location/gsInstaller/confSupport/sql/oracle/hive-schema-0.
10.0.oracle.sql
```

Instructions to deploy and configure PostgreSQL database

1. Connect to the host machine where you plan to deploy PostgreSQL instance and from a terminal window, type:

   - For RHEL and CentOS:

     ```
     yum install postgresql-server
     ```

   - For SLES:

     ```
     zypper install postgresql-server
     ```

2. Start the instance.For RHEL and CentOS:

   ```
   /etc/init.d/postgresql start
   ```

   

   Note

   For some newer versions of PostgreSQL, you might need to execute the following command:

   ```
   /etc/init.d/postgresql initdb
   ```

3. Reconfigure PostgreSQL server:

   a. Edit the `/var/lib/pgsql/data/postgresql.conf` file and change the value of **#listen_addresses = 'localhost'** to the following:

   ```
   listen_addresses = '*'
   ```

b. Edit the `/var/lib/pgsql/data/postgresql.conf` file and change the port setting **#port = 5432** to the following:

```
port = 5432
```

c. Edit the `/var/lib/pgsql/data/pg_hba.conf` and add the following:

```
host all all 0.0.0.0/0 trust
```

d. Optional - If you are using PostgreSQL v9.1 or later, add the following to the `/var/lib/pgsql/data/postgresql.conf` file:

```
standard_conforming_strings = off
```

4. Create users for PostgreSQL server:

```
echo "CREATE DATABASE $dbname;" | psql -U postgres
echo "CREATE USER $user WITH PASSWORD '$passwd';" | psql -U postgres
echo "GRANT ALL PRIVILEGES ON DATABASE $dbname TO $user;" | psql -U
 postgres
```

Note

For access to Hive metastore, create `hive_dbuser` and for access to Oozie metastore, create `oozie_dbuser`.

5. Ensure that the following SQL script is run against your Hive schema:

```
$master-install-location/gsInstaller/confSupport/sql/postgres/hive-schema-0.
10.0.postgres.sql
```

where *`$master-install-location`* is the master install location for Hive metastore host machine.

1.2.6. JDK Requirements

Your system must have the correct JDK installed on all the nodes of the cluster. HDP requires Oracle JDK 1.6 update 31.

Use the following instructions to manually install JDK 1.6 update 31:

1. Check the version. From a terminal window, type:

```
java -version
```

2. (Optional) Uninstall the Java package if the JDK version is less than v1.6 update 31.

```
rpm -qa | grep java
yum remove {java-1.*}
```

3. (Optional) Verify that the default Java package is uninstalled.

```
which java
```

4. Download the Oracle 64-bit JDK (jdk-6u31-linux-x64.bin) from the Oracle download site:

```
wget http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-
archive-downloads-javase6-419409.html#jdk-6u31-oth-JPR
```

Accept the license agreement.

5. 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
```

6. 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
```

7. 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
```

1.2.7. Virtualization and Cloud Platforms

HDP is certified and supported when running on virtual or cloud platforms (for example, VMware vSphere or Amazon Web Services EC2) as long as the respective guest operating system (OS) is supported by HDP and any issues detected on these platforms are reproducible on the same supported OS installed on bare metal.

See Operating Systems Requirements for the list of supported operating systems for HDP.

1.3. Collect Information

To deploy your HDP installation, you need to collect the following information:

- The fully qualified domain name (FQDN) for each host in your system, and which component(s) you wish to set up on which host. You can use `hostname -f` to check for the FQDN if you do not know it.
- The hostname (for an existing instance), database name, username, and password for the MySQL/Oracle instance, if you want to use external database for Hive or Oozie metastore.

Note

If you are using an existing instance, the database user you create for HDP's use must be granted `ALL PRIVILEGES` on that instance.

1.4. Decide on Deployment Type

While it is possible to deploy all of HDP on a single host, this is appropriate only for initial evaluation. In general you should use at least three hosts: one master host and two slaves.

1.5. Prepare the Environment

To deploy your HDP instance, you need to prepare your deploy environment:

- Enable NTP on the Cluster
- Check DNS
- Disable SELinux

1.5.1. Enable NTP on the Cluster

The clocks of all the nodes in your cluster must be able to synchronize with each other. If your system does not have access to the Internet, set up a master node as an NTP server.

1.5.2. Check DNS

All hosts in your system must be configured for DNS and Reverse DNS.

Note

If you are unable to configure DNS and Reverse DNS, you must edit the hosts file on every host in your cluster to contain each of your hosts.

1.5.3. Disable SELinux

SELinux can interfere with the installation process.

1.6. Create Service Users and Groups

In general Hadoop services should be owned by specific users and not by root or application users. The table below shows typical users for Hadoop services. Identify the users that you want for your Hadoop services and the common Hadoop group and create these accounts on your system.

Note

If you are considering installing your cluster in secure mode, either at installation or at a later time, you need to understand the relationship between OS system service users and Kerberos principals. Hadoop uses group memberships of users at various places, such as to determine group ownership for files or for access control. In order for Hadoop to be able to connect a Kerberos principal with its respective OS system service user, a mapping must be created. For more information on this process, see Setting Up Security for Manual Installs

Table 1.1. Typical Service Users and Groups

Hadoop Service	User	Group
HDFS	hdfs	hadoop

Hadoop Service	User	Group
MapReduce	mapred	hadoop
Hive	hive	hadoop
Pig	pig	hadoop
HCatalog/WebHCat	hcat	hadoop
HBase	hbase	hadoop
ZooKeeper	zookeeper	hadoop
Oozie	oozie	hadoop

1.7. Download Companion Files

We have provided a set of companion files, including script files (`scripts.zip`) and configuration files (`configuration_files.zip`), that you should download and use throughout this process. Download and extract the files:

```
wget http://public-repo-1.hortonworks.com/HDP/tools/1.3.0.0/
hdp_manual_install_rpm_helper_files-1.3.0.1.3.0.0-107.tar.gz
```

1.8. Define Environment Parameters

You need to set up specific users and directories for your HDP installation. Use the following instructions to define environment parameters:

1. Define Users and Groups
2. Define Directories

1.8.1. Define Users and Groups

The following table describes system user account and groups. Use this table to define what you are going to use in setting up your environment. These users and groups should reflect the accounts you created in Create System Users and Groups.

Note

The `scripts.zip` file you downloaded in Download Companion Files includes a script, `usersAndGroups.sh` for setting user and group environment parameters. We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.2. Define Users and Groups for Systems

Parameter	Definition
HDFS_USER	User owning the HDFS services. For example, `hdfs`.
MAPRED_USER	User owning the MapReduce services. For example, `mapred`.
ZOOKEEPER_USER	User owning the ZooKeeper services. For example, `zookeeper`.
HIVE_USER	User owning the Hive services. For example, `hive`.

Parameter	Definition
WEBHCAT_USER	User owning the WebHCat services. For example, hcat.
HBASE_USER	User owning the HBase services. For example, hbase.
OOZIE_USER	User owning the Oozie services. For example, oozie.
PIG_USER	User owning the Pig services. For example, pig.
HADOOP_GROUP	A common group shared by services. For example, hadoop.

1.8.2. Define Directories

The following table describes the directories for install, configuration, data, process IDs, and logs based on the Hadoop Services you plan to install. Use this table to define what you are going to use in setting up your environment.

Note

The `scripts.zip` file you downloaded in Download Companion Files includes a script, `directories.sh`, for setting directory environment parameters. We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.3. Define Directories for Core Hadoop

Hadoop Service	Parameter	Definition
HDFS	DFS_NAME_DIR	Space separated list of directories where NameNode should store the file system image. For example, /grid/hadoop/hdfs/nn /grid1/hadoop/hdfs/nn
HDFS	DFS_DATA_DIR	Space separated list of directories where DataNodes should store the blocks. For example, /grid/hadoop/hdfs/dn /grid1/hadoop/hdfs/dn /grid2/hadoop/hdfs/dn
HDFS	FS_CHECKPOINT_DIR	Space separated list of directories where SecondaryNameNode should store the checkpoint image. For example, /grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/snn /grid2/hadoop/hdfs/snn
HDFS	HDFS_LOG_DIR	Directory for storing the HDFS logs. This directory name is a combination of a directory and the $HDFS_USER.

Hadoop Service	Parameter	Definition
		For example, `/var/log/hadoop/hdfs` where `hdfs` is the `$HDFS_USER`
HDFS	`HDFS_PID_DIR`	Directory for storing the HDFS process ID. This directory name is a combination of a directory and the $HDFS_USER. For example, `/var/run/hadoop/hdfs` where `hdfs` is the `$HDFS_USER`
HDFS	`HADOOP_CONF_DIR`	Directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`

MapReduce	`MAPREDUCE_LOCAL_DIR`	Space separated list of directories where MapReduce should store temporary data.
		
		For example,
		
		`/grid/hadoop/mapred`
		
		`/grid1/hadoop/mapred`
		
		`/grid2/hadoop/mapred`

MapReduce	`MAPRED_LOG_DIR`	Directory for storing the HDFS logs. For example, `/var/log/hadoop/mapred` This directory name is a combination of a directory and the `$MAPRED_USER`. In the example `mapred` is the `$MAPRED_USER`
MapReduce	`MAPRED_PID_DIR`	Directory for storing the MapReduce process ID. For example, `/var/run/hadoop/mapred` This directory name is a combination of a directory and the `$MAPRED_USER`. In the example, `mapred` is the `$MAPRED_USER`.

Table 1.4. Define Directories for Ecosystem Components

Hadoop Service	Parameter	Definition
Pig	`PIG_CONF_DIR`	Directory to store the Pig configuration files. For example, `/etc/pig/conf`
Oozie	`OOZIE_CONF_DIR`	Directory to store the Oozie configuration files. For example, `/etc/oozie/conf`
Oozie	`OOZIE_DATA`	Directory to store the Oozie data. For example, `/var/db/oozie`
Oozie	`OOZIE_LOG_DIR`	Directory to store the Oozie logs. For example, `/var/log/oozie`
Oozie	`OOZIE_PID_DIR`	Directory to store the Oozie process ID. For example, `/var/run/oozie`

Hadoop Service	Parameter	Definition
Oozie	OOZIE_TMP_DIR	Directory to store the Oozie temporary files. For example, /var/tmp/oozie
Hive	HIVE_CONF_DIR	Directory to store the Hive configuration files. For example, /etc/hive/conf
Hive	HIVE_LOG_DIR	Directory to store the Hive logs. For example, /var/log/hive
Hive	HIVE_PID_DIR	Directory to store the Hive process ID. For example, /var/run/hive
WebHCat	WEBHCAT_CONF_DIR	Directory to store the WebHCat configuration files. For example, /etc/hcatalog/conf/webhcat
WebHCat	WEBHCAT_LOG_DIR	Directory to store the WebHCat logs. For example, /grid/0/var/log/webhcat/webhcat
WebHCat	WEBHCAT_PID_DIR	Directory to store the WebHCat process ID. For example, /var/run/webhcat
HBase	HBASE_CONF_DIR	Directory to store the HBase configuration files. For example, /etc/hbase/conf
HBase	HBASE_LOG_DIR	Directory to store the HBase logs. For example, /var/log/hbase
HBase	HBASE_PID_DIR	Directory to store the HBase process ID. For example, /var/run/hbase
ZooKeeper	ZOOKEEPER_DATA_DIR	Directory where ZooKeeper will store data. For example, /grid1/hadoop/zookeeper/data
ZooKeeper	ZOOKEEPER_CONF_DIR	Directory to store the ZooKeeper configuration files. For example, /etc/zookeeper/conf
ZooKeeper	ZOOKEEPER_LOG_DIR	Directory to store the ZooKeeper logs. For example, /var/log/zookeeper
ZooKeeper	ZOOKEEPER_PID_DIR	Directory to store the ZooKeeper process ID. For example, /var/run/zookeeper
Sqoop	SQOOP_CONF_DIR	Directory to store the Sqoop configuration files. For example, /usr/lib/sqoop/conf

2. Installing HDFS and MapReduce

Use the following instructions to install the Hadoop Core components, HDFS and MapReduce:

- Set Default File and Directory Permissions
- Install the Hadoop RPMs
- Install Compression Libraries
- Install Compression Libraries

2.1. Set Default File and Directory Permissions

Set the default file and directory permissions to 0022 (022). This is typically the default for most Linux distributions. Use the `umask` command to confirm and set as necessary. Be sure the correct `umask` is set for all terminal sessions that you use during installation.

2.2. Install the Hadoop RPMs

Execute the following command on all cluster nodes. From a terminal window, type:

- For RHEL and CentOS

```
yum install hadoop hadoop-libhdfs hadoop-native hadoop-pipes hadoop-sbin
 openssl
```

- For SLES

```
zypper install hadoop hadoop-libhdfs hadoop-native hadoop-pipes hadoop-sbin
 openssl
```

2.3. Install Compression Libraries

Make the following compression libraries available on all the cluster nodes:

1. Install Snappy.

   Complete the following instructions on all the nodes in your cluster:

   a. Install Snappy.

   - For RHEL and CentOS

```
yum install snappy snappy-devel
```

   - For SLES

```
zypper install snappy snappy-devel
```

b. Make the Snappy libraries available to Hadoop:

```
ln -sf /usr/lib64/libsnappy.so /usr/lib/hadoop/lib/native/Linux-amd64-64/
.
```

2. Install LZO.

   Execute the following command on all the nodes in your cluster. From a terminal window, type:

   - For RHEL and CentOS

     ```
     yum install hadoop-lzo lzo lzo-devel hadoop-lzo-native
     ```

   - For SLES

     ```
     zypper install lzo lzo-devel hadoop-lzo hadoop-lzo-native
     ```

2.4. Create Directories

Create directories and configure ownership + permissions on the appropriate hosts as described below. If any of these directories already exist, we recommend deleting and recreating them.

Use the following instructions to create appropriate directories:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Create the NameNode Directories
3. Create the Secondary NameNode Directories
4. Create the DataNode and MapReduce Local Directories
5. Create the Log and PID Directories

2.4.1. Create the NameNode Directories

On the node that hosts the NameNode service, execute the following commands:

```
mkdir -p $DFS_NAME_DIR
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_NAME_DIR
chmod -R 755 $DFS_NAME_DIR
```

where:

- *$DFS_NAME_DIR* is the space separated list of directories where NameNode stores the file system image. For example, `/grid/hadoop/hdfs/nn /grid1/hadoop/hdfs/nn`.

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.2. Create the SecondaryNameNode Directories

On all the nodes that can potentially host the SecondaryNameNode service, execute the following commands:

```
mkdir -p $FS_CHECKPOINT_DIR
chown -R $HDFS_USER:$HADOOP_GROUP $FS_CHECKPOINT_DIR
chmod -R 755 $FS_CHECKPOINT_DIR
```

where:

- *$FS_CHECKPOINT_DIR* is the space separated list of directories where SecondaryNameNode should store the checkpoint image. For example, `/grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/snn`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.3. Create the DataNode and MapReduce Local Directories

On all DataNodes, execute the following commands:

```
mkdir -p $DFS_DATA_DIR
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIRM
chmod -R 750 $DFS_DATA_DIR
```

On the JobTracker and all Datanodes, execute the following commands:

```
mkdir -p $MAPREDUCE_LOCAL_DIR
chown -R $MAPRED_USER:$HADOOP_GROUP $MAPREDUCE_LOCAL_DIR
chmod -R 755 $MAPREDUCE_LOCAL_DIR
```

where:

- *$DFS_DATA_DIR* is the space separated list of directories where DataNodes should store the blocks. For example, `/grid/hadoop/hdfs/dn /grid1/hadoop/hdfs/dn`.
- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$MAPREDUCE_LOCAL_DIR* is the space separated list of directories where MapReduce should store temporary data. For example, `/grid/hadoop/mapred /grid1/hadoop/mapred /grid2/hadoop/mapred`.
- *$MAPRED_USER* is the user owning the MapReduce services. For example, `mapred`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

2.4.4. Create the Log and PID Directories

On all nodes, execute the following commands:

```
mkdir -p $HDFS_LOG_DIR
chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_LOG_DIR
chmod -R 755 $HDFS_LOG_DIR
```

```
mkdir -p $MAPRED_LOG_DIR
chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_LOG_DIR
chmod -R 755 $MAPRED_LOG_DIR
```

```
mkdir -p $HDFS_PID_DIR
chown -R $HDFS_USER:$HADOOP_GROUP $HDFS_PID_DIR
chmod -R 755 $HDFS_PID_DIR
```

```
mkdir -p $MAPRED_PID_DIR
chown -R $MAPRED_USER:$HADOOP_GROUP $MAPRED_PID_DIR
chmod -R 755 $MAPRED_PID_DIR
```

where:

- *$HDFS_LOG_DIR* is the directory for storing the HDFS logs.

  This directory name is a combination of a directory and the *$HDFS_USER*. For example, `/var/log/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*.

- *$HDFS_PID_DIR* is the directory for storing the HDFS process ID.

  This directory name is a combination of a directory and the *$HDFS_USER*. For example, `/var/run/hadoop/hdfs` where `hdfs` is the *$HDFS_USER*.

- *$MAPRED_LOG_DIR* is the directory for storing the MapReduce logs.

  This directory name is a combination of a directory and the *$MAPRED_USER*. For example, `/var/log/hadoop/mapred` where `mapred` is the *$MAPRED_USER*.

- *$MAPRED_PID_DIR* is the directory for storing the MapReduce process ID.

  This directory name is a combination of a directory and the *$MAPRED_USER*. For example, `/var/run/hadoop/mapred` where `mapred` is the *$MAPRED_USER*.

3. Setting Up the Hadoop Configuration

This section describes how to set up and edit the deployment configuration files for HDFS and MapReduce.

Use the following instructions to set up Hadoop configuration files:

1. We strongly suggest that you edit and source the files included in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. From the downloaded `scripts.zip` file, extract the files from the `configuration_files/core_hadoop`directory to a temporary directory.

3. Modify the configuration files.

   In the temporary directory, locate the following files and modify the properties based on your environment. Search for TODO in the files for the properties to replace. See Define Environment Parameters for more information.

   a. Edit the `core-site.xml` file and modify the following properties:

```
<property>
 <name>fs.default.name</name>
 <value>hdfs://$namenode.full.hostname:8020</value>
 <description>Enter your NameNode hostname</description>
</property>
```

```
<property>
 <name>fs.checkpoint.dir</name>
 <value>/grid/hadoop/hdfs/snn,/grid1/hadoop/hdfs/snn,/grid2/hadoop/hdfs/
snn</value>
 <description>A comma separated list of paths. Use the list of
 directories from $FS_CHECKPOINT_DIR.
               For example, /grid/hadoop/hdfs/snn,sbr/grid1/hadoop/hdfs/
snn,sbr/grid2/hadoop/hdfs/snn </description>
</property>
```

   b. Edit the `hdfs-site.xml` file and modify the following properties:

```
<property>
 <name>dfs.name.dir</name>
 <value>/grid/hadoop/hdfs/nn,/grid1/hadoop/hdfs/nn</value>
 <description>Comma separated list of paths. Use the list of directories
 from $DFS_NAME_DIR.
               For example, /grid/hadoop/hdfs/nn,/grid1/hadoop/hdfs/nn.
</description>
</property>
```

```
<property>
 <name>dfs.data.dir</name>
 <value>/grid/hadoop/hdfs/dn,/grid1/hadoop/hdfs/dn</value>
 <description>Comma separated list of paths. Use the list of directories
 from $DFS_DATA_DIR.
               For example, /grid/hadoop/hdfs/dn,/grid1/hadoop/hdfs/dn.
</description>
</property>
```

```
<property>
 <name>dfs.http.address</name>
 <value>$namenode.full.hostname:50070</value>
 <description>Enter your NameNode hostname for http access.</description>
</property>
```

```
<property>
 <name>dfs.secondary.http.address</name>
 <value>$secondarynamenode.full.hostname:50090</value>
 <description>Enter your Secondary NameNode hostname.</description>
</property>
```

```
<property>
 <name>dfs.https.address</name>
 <value>$namenode.full.hostname:50470</value>
 <description>Enter your NameNode hostname for https access.</
description>
</property>
```

Note

The value of NameNode new generation size should be 1/8 of maximum heap size (`-Xmx`). Please check this value, as the default setting may not be accurate. To change the default value, edit the `/etc/hadoop/conf/hadoop-env.sh` file and change the value of the `-XX:MaxnewSize` parameter to 1/8th the value of the maximum heap size (`-Xmx`) parameter. Also ensure that the NameNode and Secondary NameNode have identical memory settings.

c. Edit the `mapred-site.xml` file and modify the following properties:

```
<property>
 <name>mapred.job.tracker</name>
 <value>$jobtracker.full.hostname:50300</value>
 <description>Enter your JobTracker hostname.</description>
</property>
```

```
<property>
 <name>mapred.job.tracker.http.address</name>
 <value>$jobtracker.full.hostname:50030</value>
 <description>Enter your JobTracker hostname.</description>
</property>
```

```
<property>
 <name>mapred.local.dir</name>
 <value>/grid/hadoop/mapred,/grid1/hadoop/mapred</value>
 <description>Comma separated list of paths. Use the list of directories
 from $MAPREDUCE_LOCAL_DIR</description>
</property>
```

```
<property>
 <name>mapreduce.tasktracker.group</name>
 <value>hadoop</value>
 <description>Enter your group. Use the value of $HADOOP_GROUP</
description>
</property>
```

```
<property>
 <name>mapreduce.history.server.http.address</name>
 <value>$jobtracker.full.hostname:51111</value>
 <description>Enter your JobTracker hostname</description>
</property>
```

d. Edit the `taskcontroller.cfg` file and modify the following property:

```
<property>
 <name>mapred.local.dir</name>
 <value>/grid/hadoop/mapred,/grid1/hadoop/mapred</value>
 <description>Comma separated list of paths. Use the list of directories
 from $MAPREDUCE_LOCAL_DIR</description>
</property>
```

4. Copy the configuration files.

   a. Replace the installed Hadoop configs with the modified core_hadoop configuration files and set appropriate permissions.

   ```
   rm -rf $HADOOP_CONF_DIR
   mkdir -p $HADOOP_CONF_DIR
   ```

   b. Copy all the modified configuration files in core_hadoop to *$HADOOP_CONF_DIR* on all nodes.

   c. Set appropriate permissions.

   ```
   chmod a+x $HADOOP_CONF_DIR/
   chown -R $HDFS_USER:$HADOOP_GROUP $HADOOP_CONF_DIR/../
   chmod -R 755 $HADOOP_CONF_DIR/../
   ```

   where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

4. Validating the Core Hadoop Installation

This section describes starting Core Hadoop and doing simple smoke tests. Use the following instructions to validate core Hadoop installation:

1. Format and start HDFS.

   a. Execute these commands on the NameNode:

   ```
   su $HDFS_USER
   /usr/lib/hadoop/bin/hadoop namenode -format
   /usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
    namenode
   ```

   b. Execute these commands on the Secondary NameNode :

   ```
   su $HDFS_USER
   /usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
    secondarynamenode
   ```

   c. Execute these commands on all DataNodes:

   ```
   su $HDFS_USER
   /usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
    datanode
   ```

   where:

   - *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
   - *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

2. Smoke Test HDFS.

   a. See if you can reach the NameNode server with your browser:

   ```
   http://$namenode.full.hostname:50070
   ```

   b. Try copying a file into HDFS and listing that file:

   ```
   su $HDFS_USER
   /usr/lib/hadoop/bin/hadoop dfs -copyFromLocal /etc/passwd passwd-test
   /usr/lib/hadoop/bin/hadoop dfs -ls
   ```

   c. Test browsing HDFS:

   ```
   http://$datanode.full.hostname:50075/browseDirectory.jsp?dir=/
   ```

3. Start MapReduce.

   a. Execute these commands from the JobTracker server:

   ```
   su $HDFS_USER
   ```

```
/usr/lib/hadoop/bin/hadoop fs -mkdir /mapred
/usr/lib/hadoop/bin/hadoop fs -chown -R mapred /mapred
```

```
su $MAPRED_USER
/usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 jobtracker
```

b. Execute these commands from the JobHistory server:

```
su $MAPRED_USER
/usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 historyserver
```

c. Execute these commands from all TaskTracker nodes:

```
su $MAPRED_USER
/usr/lib/hadoop/bin/hadoop-daemon.sh --config $HADOOP_CONF_DIR start
 tasktracker
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$MAPRED_USER* is the user owning the MapReduce services. For example, `mapred`.
- *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

4. Smoke Test MapReduce.

a. Try browsing to the JobTracker:

```
http://$jobtracker.full.hostname:50030/
```

b. Smoke test using Teragen (to generate 10GB of data) and then using Terasort to sort the data.

```
sus $HDFS_USER
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop/hadoop-examples.jar
 teragen 100000000 /test/10gsort/input
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop/hadoop-examples.jar
 terasort /test/10gsort/input /test/10gsort/output
```

5. Installing Apache Pig

This section describes installing and testing Apache Pig, a platform for creating higher level data flow programs that can be compiled into sequences of MapReduce programs, using Pig Latin, the platform's native language.

Complete the following instructions to install Pig:

1. Install the Pig RPMs
2. Set up configuration files
3. Validate the installation

5.1. Install the Pig RPMs

On all hosts on which Pig programs will be executed, install the RPMs.

- For RHEL/CentOS

```
yum install pig
```

- For SLES

```
zypper install pig
```

5.2. Set Up Configuration Files

There are several configuration files that need to be set up for Pig.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Pig configuration files:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files). Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.
2. From the file you downloaded in extract the files in `configuration_files/pig` directory to a temporary directory.
3. Copy the configuration files.

   On all hosts where Pig will be executed, replace the installed Pig configs with the downloaded one and set appropriate permissions:

```
rm -rf $PIG_CONF_DIR
mkdir -p $PIG_CONF_DIR
```

   <Copy the all config files to *$PIG_CONF_DIR*>

```
chmod -R 755 $PIG_CONF_DIR/../
```

where:

- *$PIG_CONF_DIR* is the directory to store the Pig logs. For example, `/etc/pig/conf`.
- *$PIG_USER* is the user owning the Pig services. For example, `pig`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

5.3. Validate the Installation

Use the following steps to validate your installation:

1. Use a terminal window on a machine where Pig is installed and execute the following commands :

```
login as $HDFS_USER
/usr/lib/hadoop/bin/hadoop dfs -copyFromLocal /etc/passwd passwd
```

2. Execute the following commands to produce script file `/tmp/id.pig`:

```
echo "A = load 'passwd' using PigStorage(':'); " > /tmp/id.pig
echo "B = foreach A generate \$0 as id; store B into '/tmp/id.out'; " >> /
tmp/id.pig
```

3. Execute the Pig script:

```
pig -l /tmp/pig.log /tmp/id.pig
```

6. Installing Apache Hive and Apache HCatalog

This section describes installing and testing Apache Hive, a tool for creating higher level SQL queries using HiveQL, the tool's native language that can then be compiled into sequences of MapReduce programs. It also describes installing and testing Apache HCatalog, a metadata abstraction layer that insulates users and scripts from how and where data is physically stored.

Complete the following instructions to install Hive and HCatalog:

1. Install the Hive and HCatalog RPMs
2. Set Directories and Permissions
3. Set Up the Hive/HCatalog Configuration Files
4. Validate the Installation

6.1. Install the Hive and HCatalog RPMs

1. On all Hive client/gateway nodes (on which Hive programs will be executed), Hive Metastore Server, and HiveServer2 machine, install the Hive RPMs.
   - For RHEL/CentOS:

     ```
     yum install hive hcatalog
     ```

   - For SLES:

     ```
     zypper install hive hcatalog
     ```

2. Optional: Download and add the database connector JAR.
   - **For MySQL:**
     a. Execute the following command on the Hive metastore machine.
        - RHEL/CentOS:

          ```
          yum install mysql-connector-java
          ```

        - For SLES:

          ```
          zypper install mysql-connector-java
          ```

     b. Unzip and copy the downloaded JAR file the `/usr/lib/hive/lib/` directory on your Hive host machine.

     c. Ensure that the JAR file has appropriate permissions.

   - **For Oracle:** Note that these instructions are for OJDBC driver for Oracle 11g.

a. On the Hive metastore host machine, download the Oracle JDBC (OJDBC) driver from here.

b. Copy the JAR file to *$HIVE_HOME*/lib/.

$HIVE_HOME is by default configured to usr/lib/hive.

c. Ensure that the JAR file has appropriate permissions.

- **For PostgreSQL:**

a. Execute the following command on the Hive metastore machine.

- RHEL/CentOS:

```
yum install postgresql-jdbc
```

- For SLES:

```
zypper install postgresql-jdbc
```

b. Execute the following command on the Hive metastore machine:

```
ln -sf /usr/share/java/postgresql-jdbc.jar $HIVE_HOME/lib/.
```

where *$HIVE_HOME* is by default configured to usr/lib/hive.

c. Ensure that the JAR file has appropriate permissions.

6.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Hive and HCatalog configuration files :

1. We strongly suggest that you edit and source the files included in scripts.zip file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your ~/.bash_profile) to set up these environment variables in your environment.

2. Execute these commands on the Hive server machine:

```
mkdir -p $HIVE_LOG_DIR;
chown -R $HIVE_USER:$HADOOP_GROUP $HIVE_LOG_DIR;
chmod -R 755 $HIVE_LOG_DIR;
```

   where:

   - *$HIVE_LOG_DIR* is the directory for storing theHive Server logs.

     This directory name is a combination of a directory and the *$HIVE_USER*.

- *$HIVE_USER* is the user owning the Hive services. For example, `hive`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6.3. Set Up the Hive/HCatalog Configuration Files

There are several configuration files that need to be set up for Hive/HCatalog.

In the temporary directory, locate the following file and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

Use the following instructions to set up the Hive/HCatalog configuration files:

1. Extract the Hive/HCatalog configuration files.

   From the downloaded `scripts.zip` file, extract the files in `configuration_files/hive` directory to a temporary directory.

2. Modify the configuration files.

   In the temporary directory, locate the following file and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

   a. Edit `hive-site.xml` and modify the following properties:

```
<property>
 <name>javax.jdo.option.ConnectionURL</name>
 <value></value>
 <description>Enter your JDBC connection string.
                For MySQL database: jdbc:mysql://$mysql.full.
hostname:3306/$database.name?createDatabaseIfNotExist=true
                For Oracle
 database: jdbc:oracle:thin:@$dbhost:1521/$hive_dbname
                For PostgreSQL database: jdbc:postgresql:/
/$dbhost:5432/$hive_dbname
 </description>
</property>
```

```
<property>
 <name>javax.jdo.option.ConnectionDriverName</name>
 <value></value>
 <description>JDBC Connection Driver Name.
                For MySQL database: com.mysql.jdbc.Driver
                For Oracle database: oracle.jdbc.driver.OracleDriver
                For PostgreSQL database: org.postgresql.Driver

</property>
```

```
<property>
 <name>javax.jdo.option.ConnectionUserName</name>
 <value>$dbusername</value>
 <description>Enter your MySQL/Oracle/PostgreSQL credentials. </
description>
</property>
```

```
<property>
 <name>javax.jdo.option.ConnectionPassword</name>
 <value>$dbuserpassword</value>
 <description>Enter your MySQL/Oracle/PostgreSQL credentials. </
description>
</property>
```

```
<property>
 <name>hive.metastore.uris</name>
 <value>thrift://$metastore.server.full.hostname:9083</value>
 <description>URI for client to contact metastore server. To enable
 HiveServer2, leave the property value empty. </description>
</property>
```

If using PostgreSQL server, add the following properties:

```

<property>
 <name>datanucleus.autoCreateSchema</name>
 <value>false</value>

</property>
```

3. Copy the configuration files.

   a. On all Hive hosts create the Hive configuration directory.

   ```
   rm -r $HIVE_CONF_DIR ;
   mkdir -p $HIVE_CONF_DIR ;
   ```

   b. Copy all the configuration files to *$HIVE_CONF_DIR* directory.

   c. Set appropriate permissions:

   ```
   chown -R $HIVE_USER:$HADOOP_GROUP $HIVE_CONF_DIR/../ ;
   chmod -R 755 $HIVE_CONF_DIR/../ ;
   ```

   where:

   - *$HIVE_CONF_DIR* is the directory to store the Hive configuration files. For example, `/etc/hive/conf`.
   - *$HIVE_USER* is the user owning the Hive services. For example, `hive`.
   - *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6.4. Validate the Installation

Use these steps to validate your installation.

1. Start Hive Metastore service.

   a. Start your metastore database server.

   - For PostgreSQL:

   ```
   psql -U postgres -f $Path_to_PostgreSQL_Script $hive_dbname
   ```

b. Start Hive Metastore service.

```
Login as $HIVE_USER
nohup hive --service metastore>$HIVE_LOG_DIR/hive.out 2>$HIVE_LOG_DIR/
hive.log &
```

2. Smoke Test Hive.

a. Open Hive command line shell.

```
hive
```

b. Run sample commands.

```
show databases;
create table test(col1 int, col2 string);
show tables;
```

3. Start HiveServer2.

```
 /usr/lib/hive/bin/hiveserver2 -hiveconf hive.metastore.uris=" " >
$HIVE_LOG_DIR/hiveserver2.out 2> $HIVE_LOG_DIR/hiveserver2.log &
```

4. Smoke Test HiveServer2.

a. Open Beeline command line shell to interact with HiveServer2.

```
/usr/lib/hive/bin/beeline
```

b. Establish connection to server.

```
!connect jdbc:hive2://$hive.server.full.hostname:10000 $HIVE_USER
 password org.apache.hive.jdbc.HiveDriver
```

c. Run sample commands.

```
show databases;
create table test2(a int, b string);
show tables;
```

7. Installing WebHCat

This section describes installing and testing WebHCat, which provides a REST interface to Apache HCatalog services like job submission and eventing.

Use the following instructions to install WebHCat:

1. Install the WebHCat RPMs
2. Set Directories and Permissions
3. Modify WebHCat Configuration Files
4. Set Up HDFS User and Prepare WebHCat Directories On HDFS
5. Validate the Installation

7.1. Install the WebHCat RPMs

On the WebHCat server machine, install the necessary RPMs.

- For RHEL/CentOS:

```
yum install hcatalog webhcat-tar-hive webhcat-tar-pig
```

- For SLES:

```
zypper install hcatalog webhcat-tar-hive webhcat-tar-pig
```

7.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Pig configuration files :

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute these commands on your WebHCat server machine to create log and pid directories.

```
mkdir -p $WEBHCAT_LOG_DIR
chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_LOG_DIR
chmod -R 755 $WEBHCAT_LOG_DIR
```

```
mkdir -p $WEBHCAT_PID_DIR
chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_PID_DIR
```

```
chmod -R 755 $WEBHCAT_PID_DIR
```

where:

- *$WEBHCAT_LOG_DIR* is the directory to store the WebHCat logs. For example, `/grid/0/var/log/webhcat/webhcat`.
- *$WEBHCAT_PID_DIR* is the directory to store the WebHCat process ID. For example, `/var/run/webhcat`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

7.3. Modify WebHCat Config Files

Use the following instructions to modify the WebHCat config files:

1. Extract the WebHCat configuration files

   From the downloaded `scripts.zip` file, extract the files in `configuration_files/webhcat` directory to a temporary location.

2. Modify the configuration files

   In the temporary directory, locate the following files and modify the properties based on your environment.

   Search for `TODO` in the files for the properties to replace. See Define Environment Parameters for more information.

   a. Edit the `webhcat-site.xml` and modify the following properties:

   ```
   <property>
    <name>templeton.hive.properties</name>
    <value>hive.metastore.local=false, hive.metastore.uris=thrift:/
   /$metastore.server.full.hostname:9083,hive.metastore.sasl.enabled=no,
   hive.metastore.execute.setugi=true</value>
    <description>Properties to set when running Hive.</description>
   </property>
   ```

   ```
   <property>
    <name>templeton.zookeeper.hosts</name>
    <value>$zookeeper1.full.hostname:2181,$zookeeper1.full.hostname:2181,..
   </value>
    <description>ZooKeeper servers, as comma separated HOST:PORT pairs.</
   description>
   </property>
   ```

   ```
   <property>
    <name>hive.metastore.warehouse.dir</name>
    <value>/path/to/warehouse/dir</value>
   </property>
   ```

3. Set up the WebHCat configuration files.

   a. Delete any existing WebHCat configuration files:

```
rm -rf  $WEBHCAT_CONF_DIR/*
```

b. Copy all the config files to *$WEBHCAT_CONF_DIR* and set appropriate permissions:

```
chown -R $WEBHCAT_USER:$HADOOP_GROUP $WEBHCAT_CONF_DIR
chmod -R 755 $WEBHCAT_CONF_DIR
```

where:

- *$WEBHCAT_CONF_DIR* is the directory to store theWebHCat configuration files. For example, `/etc/hcatalog/conf/webhcat`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

7.4. Set Up the HDFS User and Prepare WebHCat Directories On HDFS

1. Set up the HDFS user. Login as *$HDFS_USER*

```
 hadoop fs -mkdir /user/$WEBHCAT_USER
hadoop fs -chown -R $WEBHCAT_USER:$WEBHCAT_USER /user/$WEBHCAT_USER
hadoop fs -mkdir /apps/webhcat
```

2. Prepare WebHCat directories on HDFS.

```
hadoop dfs -copyFromLocal /usr/share/HDP-webhcat/pig.tar.gz /apps/webhcat/
hadoop dfs -copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /apps/webhcat/
hadoop dfs -copyFromLocal /usr/lib/hadoop/contrib/streaming/hadoop-
streaming*.jar /apps/webhcat/
```

3. Set appropriate permissions for the HDFS user and the webhcat directory.

```
hadoop fs -chown -R $WEBHCAT_USER:users /apps/webhcat
hadoop fs -chmod -R 755 /apps/webhcat
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$WEBHCAT_USER* is the user owning the WebHCat services. For example, `hcat`.

7.5. Validate the Installation

1. Start the WebHCat server.

```
<login as $WEBHCAT_USER>
/usr/lib/hcatalog/sbin/webhcat_server.sh start
```

2. From the browser, type:

```
http://$WebHCat.server.full.hostname:50111/templeton/v1/status
```

You should see the following output:

```
{"status":"ok","version":"v1"}
```

8. Installing HBase and ZooKeeper

This section describes installing and testing Apache HBase, a distributed, column-oriented database that provides the ability to access and manipulate data randomly in the context of the large blocks that make up HDFS. It also describes installing and testing Apache ZooKeeper, a centralized tool for providing services to highly distributed systems.

Use the following steps to install HBase and ZooKeeper:

- Install the HBase and ZooKeeper RPMs
- Set directories and permissions
- Set up the configuration files
- Validate the installation

8.1. Install the HBase and ZooKeeper RPMs

1. Execute the following command on Zookeeper nodes and the gateway node:
   - For RHEL/CentOS:

     ```
     yum install zookeeper
     ```

   - For SLES:

     ```
     zypper install zookeeper
     ```

2. Execute the following command on HBaseMaster node, RegionServer nodes and the gateway node:
   - For RHEL/CentOS:

     ```
     yum install hbase
     ```

   - For SLES:

     ```
     zypper install hbase
     ```

8.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to create appropriate directories:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute the following commands on all nodes:

```
mkdir -p $HBASE_LOG_DIR;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_LOG_DIR;
chmod -R 755 $HBASE_LOG_DIR;
```

```
mkdir -p $HBASE_PID_DIR;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_PID_DIR;
chmod -R 755 $HBASE_PID_DIR;
```

```
mkdir -p $ZOOKEEPER_LOG_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_LOG_DIR;
chmod -R 755 $ZOOKEEPER_LOG_DIR;
```

```
mkdir -p $ZOOKEEPER_PID_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_PID_DIR;
chmod -R 755 $ZOOKEEPER_PID_DIR;
```

```
mkdir -p $ZOOKEEPER_DATA_DIR;
chmod -R 755 $ZOOKEEPER_DATA_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_DATA_DIR
```

where:

- *$HBASE_LOG_DIR* is the directory to store the HBase logs. For example, `/var/log/hbase`.
- *$HBASE_PID_DIR* is the directory to store the HBase process ID. For example, `/var/run/hbase`.
- *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.
- *$ZOOKEEPER_LOG_DIR* is the directory to store the ZooKeeper logs. For example, `/var/log/zookeeper`.
- *$ZOOKEEPER_PID_DIR* is the directory to store the ZooKeeper process ID. For example, `/var/run/zookeeper`.
- *$ZOOKEEPER_DATA_DIR* is the directory where ZooKeeper will store data. For example, `/grid1/hadoop/zookeeper/data`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

8.3. Set Up the Configuration Files

There are several configuration files that need to be set up for HBase and ZooKeeper.

- Extract the HBase and ZooKeeper configuration files.

  From the downloaded `scripts.zip` file, extract the files in `configuration_files/hbase` and `configuration_files/zookeeper` directory to separate temporary directories.

- Modify the configuration files.

In the respective temporary directories, locate the following files and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

1. Edit the `zoo.cfg` and modify the `server.1`, `server.2`, and `server.3` properties:

```
#The number of milliseconds of each tick
tickTime=2000

#The number of ticks that the initial synchronization phase can take
initLimit=10

#The number of ticks that can pass between sending a request and getting
 an acknowledgement
syncLimit=5

#The directory where the snapshot is stored.
dataDir=/hadoop/zookeeper

#The port at which the clients will connect
clientPort=2181

server.1=$zk.server1.full.hostname:2888:3888
server.2=$zk.server2.full.hostname:2888:3888
server.3=$zk.server3.full.hostname:2888:3888
```

2. Edit the `hbase-site.xml` and modify the following properties:

```
<property>
 <name>hbase.rootdir</name>
 <value>hdfs://$hbase.namenode.full.hostname:8020/apps/hbase/data</value>
 <description>Enter the HBase NameNode server hostname</description>
</property>
```

```
<property>
 <name>hbase.master.info.bindAddress</name>
 <value>0.0.0.0</value>
 <description>The bind address for the HBase Master web UI.</description>
</property>
```

```
<property>
 <name>hbase.zookeeper.quorum</name>
 <value>$zk.server1.full.hostname,$zk.server2.full.hostname,$zk.server3.
full.hostname</value>
 <description>Comma separated list of Zookeeper servers (match to what is
 specified in zoo.cfg but without portnumbers)</description>
</property>
```

- Copy the configuration files

1. On all hosts create the config directory:

```
rm -r $HBASE_CONF_DIR ;
mkdir -p $HBASE_CONF_DIR ;
```

```
rm -r $ZOOKEEPER_CONF_DIR ;
mkdir -p $ZOOKEEPER_CONF_DIR ;
```

2. Copy all the HBase configuration files to *$HBASE_CONF_DIR* and the ZooKeeper configuration files to *$ZOOKEEPER_CONF_DIR* directory.

3. Set appropriate permissions:

```
chmod a+x $HBASE_CONF_DIR/;
chown -R $HBASE_USER:$HADOOP_GROUP $HBASE_CONF_DIR/../  ;
chmod -R 755 $HBASE_CONF_DIR/../
```

```
chmod a+x $ZOOKEEPER_CONF_DIR/;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOOKEEPER_CONF_DIR/../  ;
chmod -R 755 $ZOOKEEPER_CONF_DIR/../
```

where:

- *$HBASE_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`.
- *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
- *$ZOOKEEPER_CONF_DIR* is the directory to store the ZooKeeper configuration files. For example, `/etc/zookeeper/conf`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.

8.4. Validate the Installation

Use these steps to validate your installation.

1. Start HBase and ZooKeeper.

   a. Execute this command from the each ZooKeeper node:

```
<login as $ZOOKEEPER_USER>
/usr/lib/zookeeper/bin/zkServer.sh start $ZOOKEEPER_CONF_DIR/zoo.cfg
```

   b. Execute this command from the HBase Master node:

```
<login as $HBASE_USER>
/usr/lib/hadoop/bin/hadoop fs -mkdir /apps/hbase
/usr/lib/hadoop/bin/hadoop fs -chown -R hbase /apps/hbase
/usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR start master
```

   c. Execute this command from each HBase Region Server node:

```
<login as $HBASE_USER>
/usr/lib/hbase/bin/hbase-daemon.sh --config $HBASE_CONF_DIR start
 regionserver
```

   where:

   - *$HBASE_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`.

- *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
- *$ZOOKEEPER_CONF_DIR* is the directory to store the ZooKeeper configuration files. For example, `/etc/zookeeper/conf`.
- *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.

2. Smoke Test HBase and ZooKeeper.

   From a terminal window, enter:

```
echo "echo status | hbase shell" > /tmp/hbasesmoke.sh
echo "echo disable\'usertable\' | hbase shell" >> /tmp/hbasesmoke.sh
echo "echo drop \'usertable\' | hbase shell" >> /tmp/hbasesmoke.sh
echo "echo create \'usertable\', \'family\' | hbase shell" >> /tmp/
hbasesmoke.sh
echo "echo put \'usertable\', \'row01\',\'family:col01\', \'value1\' | hbase
 shell" >> /tmp/hbasesmoke.sh
echo "echo scan\'usertable\' | hbase shell" >> /tmp/hbasesmoke.sh
```

9. Installing Apache Oozie

This section describes installing and testing Apache Oozie, a server based workflow engine optimized for running workflows that execute Hadoop jobs.

Complete the following instructions to install Oozie:

1. Install the Oozie RPMs
2. Set Directories and Permissions
3. Set Up the Oozie Configuration Files
4. Validate the Installation

9.1. Install the Oozie RPMs

1. On Oozie server, install the necessary RPMs.

```
yum install oozie extjs-2.2-1
```

2. Add the ExtJS library to the Oozie application.

```
/usr/lib/oozie/bin/oozie-setup.sh -hadoop 0.20.200 /usr/lib/hadoop -extjs /
usr/share/HDP-oozie/ext-2.2.zip
```

3. Add LZO JAR files.

```
/usr/lib/oozie/bin/oozie-setup.sh -hadoop 2.x /usr/lib/hadoop -extjs /usr/
share/HDP-oozie/ext-2.2.zip -jars /usr/lib/hadoop/lib/hadoop-lzo-0.5.0.jar
```

4. Optional: Download and add the database connector JAR.

   - **For MySQL:**

     a. Execute the following command on the Oozie metastore machine:

        - For RHEL/CentOS:

```
yum install mysql-connector-java
```

        - For SLES:

```
zypper install mysql-connector-java
```

     b. Execute the following command on your Oozie metastore machine:

```
/usr/lib/oozie/bin/oozie-setup.sh -hadoop 2.x /usr/lib/hadoop -extjs /
usr/share/HDP-oozie/ext-2.2.zip -jars /usr/lib/hadoop/lib/hadoop-lzo-0.
5.0.jar:/usr/share/java/mysql-connector-java.jar
```

     c. Ensure that the JAR file has appropriate permissions.

   - **For Oracle:** Note that the following instructions are for OJDBC driver for Oracle 11g.

     a. Download the Oracle JDBC (OJDBC) driver from here.

b. Copy the JAR file to `/usr/lib/oozie/libtools/`.

c. Ensure that the JAR file has appropriate permissions.

- **For PostgreSQL:**

a. Execute the following command on the Oozie metastore machine:

- For RHEL/CentOS:

```
yum install postgresql-jdbc
```

- For SLES:

```
zypper install postgresql-jdbc
```

b. Copy the downloaded JAR file to *$OOZIE_HOME*`/lib` directory.

$OOZIE_HOME is by default set to `/usr/lib/oozie/`.

c. Ensure that the JAR file has appropriate permissions.

9.2. Set Directories and Permissions

Create directories and configure ownership + permissions on the appropriate hosts as described below.

If any of these directories already exist, we recommend deleting and recreating them. Use the following instructions to set up Oozie configuration files:

1. We strongly suggest that you edit and source the files included in `scripts.zip` file (downloaded in Download Companion Files).

   Alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

2. Execute the following commands on your Oozie server:

```
mkdir -p $OOZIE_DATA;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_DATA;
chmod -R 755 $OOZIE_DATA;
```

```
mkdir -p $OOZIE_LOG_DIR;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_LOG_DIR;
chmod -R 755 $OOZIE_LOG_DIR;
```

```
mkdir -p $OOZIE_PID_DIR;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_PID_DIR;
chmod -R 755 $OOZIE_PID_DIR;
```

```
mkdir -p $OOZIE_TMP_DIR;
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_TMP_DIR;
chmod -R 755 $OOZIE_TMP_DIR;
```

where:

- *$OOZIE_DATA* is the directory to store the Oozie data. For example, `/var/db/oozie`.
- *$OOZIE_LOG_DIR* is the directory to store the Oozie logs. For example, `/var/log/oozie`.
- *$OOZIE_PID_DIR* is the directory to store the Oozie process ID. For example, `/var/run/oozie`.
- *$OOZIE_TMP_DIR* is the directory to store the Oozie temporary files. For example, `/var/tmp/oozie`.
- *$OOZIE_USER* is the user owning the Oozie services. For example, `oozie`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

9.3. Set Up the Oozie Configuration Files

Complete the following instructions to set up Oozie configuration files:

1. Extract the Oozie configuration files.

   From tthe downloaded `scripts.zip` file, extract the files from the `configuration_files/oozie` directory to a temporary directory.

2. Modify the configuration files.

   In the temporary directory, locate the following file and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

   a. Edit the `oozie-site.xml` and modify the following properties:

```
<property>
 <name>oozie.base.url</name>
 <value>http://$oozie.full.hostname:11000/oozie</value>
 <description>Enter your Oozie server hostname.</description>
</property>
```

```
<property>
 <name>oozie.service.StoreService.jdbc.url</name>
 <value>jdbc:derby:$OOZIE_DATA_DIR/$soozie.db.schema.name-db;create=
true</value>
 <description>JDBC URL
                For Derby database: jdbc:derby:$OOZIE_DATA_DIR/$soozie.
db.schema.name-db;create=true
                For MySQL database: jdbc:mysql://$dbhost:3306/$dbname
                For Oracle
 database: jdbc:oracle:thin:@$dbhost:1521/$oozie_dbname
                For PostgreSQL database: jdbc:postgresql:/
/$dbhost:5432/$oozie_dbname

 </description>
</property>
```

   where *$soozie.db.schema.name-db* is set to `oozie`.

```
<property>
  <name>oozie.service.StoreService.jdbc.driver</name>
     <value>org.apache.derby.jdbc.EmbeddedDriver</value>
     <description>
     JDBC driver class.
                 For MySQL database: com.mysql.jdbc.Driver
                 For Oracle database: oracle.jdbc.driver.OracleDriver
                 For PostgreSQL database: org.postgresql.Driver
     </description>
</property>
```

```
<property>
     <name>oozie.service.StoreService.jdbc.username</name>
     <value>$oozie_user</value>
     <description>
     DB user name.
     </description>
   </property>
```

```
   <property>
     <name>oozie.service.StoreService.jdbc.password</name>
     <value>$oozie_password</value>
     <description>
     DB user password.
     IMPORTANT: if password is emtpy leave a 1 space string, the service
 trims the value,
      if empty Configuration assumes it is NULL.
     IMPORTANT: if the StoreServicePasswordService is active, it will
 reset this value with the value given in
     the console.
    </description>
   </property>
```

b. Edit the `oozie-env.sh` and modify the following properties:

```
<property>
 <name>OOZIE_LOG_DIR</name>
 <value>/var/log/oozie</value>
 <description>Use value from $OOZIE_LOG_DIR </description>
</property>
```

```
<property>
 <name>OOZIE_PID_DIR</name>
 <value>/var/run/oozie</value>
 <description>Use value from $OOZIE_PID_DIR </description>
</property>
```

```
<property>
 <name>OOZIE_DATA_DIR</name>
 <value>/var/db/oozie</value>
 <description>Use value from $OOZIE_DATA_DIR </description>
</property>
```

3. Copy the Configuration Files

   On your Oozie server create the config directory, copy the config files and set the permissions:

```
rm -r $OOZIE_CONF_DIR ;
```

```
mkdir -p $OOZIE_CONF_DIR ;
```

4. Copy all the config files to *$OOZIE_CONF_DIR* directory.
5. Set appropriate permissions.

```
chown -R $OOZIE_USER:$HADOOP_GROUP $OOZIE_CONF_DIR/../ ;
chmod -R 755 $OOZIE_CONF_DIR/../ ;
```

where:

- *$OOZIE_CONF_DIR* is the directory to store Oozie configuration files. For example, `/etc/oozie/conf`.
- *$OOZIE_DATA* is the directory to store the Oozie data. For example, `/var/db/oozie`.
- *$OOZIE_LOG_DIR* is the directory to store the Oozie logs. For example, `/var/log/oozie`.
- *$OOZIE_PID_DIR* is the directory to store the Oozie process ID. For example, `/var/run/oozie`.
- *$OOZIE_TMP_DIR* is the directory to store the Oozie temporary files. For example, `/var/tmp/oozie`.
- *$OOZIE_USER* is the user owning the Oozie services. For example, `oozie`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

9.4. Validate the Installation

Use these steps to validate your installation.

1. Start the Oozie server:

```
mkdir /etc/oozie/conf/action-conf
<login as $OOZIE_USER>
/usr/lib/oozie/bin/oozie-start.sh
```

2. Confirm that you can browse to the Oozie server:

```
http://{oozie.full.hostname}:11000/oozie
```

3. Access the Oozie Server with the Oozie client.

```
oozie admin -oozie http://$oozie.full.hostname:11000/oozie -status
```

You should see the following output:

```
System mode: NORMAL
```

10. Installing Apache Sqoop

This section describes installing and testing Apache Sqoop, a component that provides a mechanism for moving data between HDFS and external structured datastores.

Use the following instructions to install Sqoop:

1. Install the Sqoop RPMs
2. Optional - Download database connector
3. Set up the Sqoop configuration
4. Validate the installation

10.1. Install the Sqoop RPMs

1. Install Sqoop RPMs.

   On all nodes where you plan to use the Sqoop client, install the RPMs:

   - For RHEL/CentOS:

     ```
     yum install sqoop
     ```

   - For SLES:

     ```
     zypper install sqoop
     ```

2. Optional: Download and add database connector.

   If you plan to migrate data from HDFS/Hive/HBase to database, you must have appropriate database connector (MySQL/Oracle/PostgreSQL) JAR file.

   - **For MySQL:**

     a. Execute the following command on the Sqoop host machine:

     - For RHEL/CentOS:

       ```
       yum install mysql-connector-java
       ```

     - For SLES:

       ```
       zypper install mysql-connector-java
       ```

     b. Copy the JAR file to *$SQOOP_HOME*/`lib`.

     $SQOOP_HOME is by default set to `/usr/lib/sqoop/`.

     c. Ensure that the JAR file has appropriate permissions.

   - **For Oracle:** Note that the following instructions are for OJDBC driver for Oracle 11g.

a. Download the Oracle JDBC (OJDBC) driver from here.

b. Copy the JAR file to *$SQOOP_HOME*/lib.

$SQOOP_HOME is by default set to /usr/lib/sqoop/.

c. Ensure that the JAR file has appropriate permissions.

- **For PostgreSQL:**

a. Execute the following command on the Sqoop host machine:

- For RHEL/CentOS:

```
yum install postgresql-jdbc
```

- For SLES:

```
zypper install postgresql-jdbc
```

b. Copy the downloaded JAR file to *$SQOOP_HOME*/lib directory.

$SQOOP_HOME is by default set to /usr/lib/sqoop/.

c. Ensure that the JAR file has appropriate permissions.

10.2. Optional - Download Database Connector

If you plan to migrate data from HDFS/Hive/HBase to database, you must have appropriate database connector (MySQL/Oracle) JAR file.

Use the following instructions to add appropriate database connector:

1. Complete the instructions listed here: Minimum requirements - Database requirements
2. Copy the JAR file to /usr/lib/sqoop/lib.

10.3. Set Up the Sqoop Configuration

There are several configuration files that need to be set up for Sqoop. Use the following instruction to set up Sqoop configurations:

1. Extract the Sqoop configuration files.

   From the downloaded scripts.zip file (downloaded in Download Companion Files), extract the files in configuration_files/sqoop directory to a temporary location.

2. Copy the configuration files to *$SQOOP_CONF_DIR* directory.

10.4. Validate the Installation

Use this step to validate your installation.

Execute the following command. You should see the Sqoop version information displayed.

```
sqoop version
```

11. Installing Ganglia

This section describes installing and testing Ganglia, a system for monitoring and capturing metrics from services and components of the Hadoop cluster.

Use the following instructions to install Ganglia:

- Install the Ganglia RPMs
- Install the configuration files
- Validate the installation

11.1. Install the Ganglia RPMs

1. On the host you have chosen to be the Ganglia server, install the server RPMs:
   - For RHEL/CentOS:

     ```
     yum install ganglia-gmond-3.2.0-99 ganglia-gmetad-3.2.0-99 gweb-2.2.0-99
      hdp_mon_ganglia_addons
     ```

   - For SLES:

     ```
     zypper install ganglia-gmond-3.2.0-99 ganglia-gmetad-3.2.0-99 gweb-2.2.
     0-99 hdp_mon_ganglia_addons
     ```

2. On each host in the cluster, install the client RPMs:
   - For RHEL/CentOS:

     ```
     yum install ganglia-gmond-3.2.0-99
     ```

   - For SLES:

     ```
     zypper install ganglia-gmond-3.2.0-99
     ```

11.2. Install the Configuration Files

There are several configuration files that need to be set up for Ganglia. Use the following instructions ot install the configuration files for Ganglia:

1. Extract the Ganglia configuration files.

   From the downloaded `scripts.zip` file (downloaded in Download Companion Files), copy the files in the `configuration_files/ganglia` directory to a temporary directory.

   The `ganglia` directory contains two sub-directories, `objects` and `scripts`.

2. Copy the configuration files.

   On the Ganglia server host, complete the following instructions:

a. Create a directory for the `objects` directory and copy the `objects` files:

```
mkdir -p /usr/libexec/hdp/ganglia
cp $tmp-directory/ganglia/objects/* /usr/libexec/hdp/ganglia
```

b. Copy the contents of the `scripts` directory to `init.d` directory.

```
cp $tmp-directory/ganglia/scripts/* /etc/init.d
```

3. On each host in the cluster, copy the Ganglia monitoring init script to `init.d` directory:

```
cp $tmp-directory/ganglia/scripts/hdp-gmond /etc/init.d
```

4. Set up Ganglia hosts.

a. On the Ganglia server, execute the following commands to configure the `gmond` collector:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPJobTracker -m
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPNameNode -m
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPSlaves -m
/usr/libexec/hdp/ganglia/setupGanglia.sh -t
```

b. If HBase is installed, execute the following command on the HBase Master host machine:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPHBaseMaster -m
```

c. On the NameNode and SecondaryNameNode servers, execute the following command to configure the `gmond` emitters:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPNameNode
```

d. On the JobTracker server, execute the following command to configure the `gmond` emitters:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPJobTracker
```

e. On all hosts, execute the following command to configure the `gmond` emitters:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPSlaves
```

f. If HBase is installed, execute the following command on the HBase Master host machine to configure the `gmond` emitter:

```
/usr/libexec/hdp/ganglia/setupGanglia.sh -c HDPHBaseMaster
```

5. Set up configurations.

a. On the Ganglia server, confirm that the `bind` property in each of the following files is set to the Ganglia server hostname:

```
/etc/ganglia/hdp/HDPNameNode/conf.d/gmond.master.conf
/etc/ganglia/hdp/HDPJobTracker/conf.d/gmond.master.conf
/etc/ganglia/hdp/HDPSlaves/conf.d/gmond.master.conf
```

And if HBase is installed:

```
/etc/ganglia/hdp/HDPHBaseMaster/conf.d/gmond.master.conf
```

b. On the Ganglia server, open the `/etc/ganglia/hdp/gmetad.conf` file and confirm that the `data_source` properties are set to the Ganglia server hostname.

For example:

```
data_source "HDPSlaves" $my.ganglia.server.hostname:8660
data_source "HDPNameNode" $my.ganglia.server.hostname:8661
data_source "HDPJobTracker" $my.ganglia.server.hostname:8662
```

And if HBase is installed:

```
data_source "HDPHBaseMaster" $my.ganglia.server.hostname:8663
```

c. On all hosts except the Ganglia server, open the slave configuration files and confirm that the `host` property is set to the Ganglia Server hostname:

```
/etc/ganglia/hdp/HDPNameNode/conf.d/gmond.slave.conf
/etc/ganglia/hdp/HDPJobTracker/conf.d/gmond.slave.conf
/etc/ganglia/hdp/HDPSlaves/conf.d/gmond.slave.conf
```

And if HBase is installed:

```
/etc/ganglia/hdp/HDPHBaseMaster/conf.d/gmond.slave.conf
```

6. Set up Hadoop metrics. On each host in the cluster, complete the following instructions:

a. Stop the Hadoop services using the instructions provided here.

b. Change to the Hadoop configuration directory.

```
cd $HADOOP_CONF_DIR
```

where *$HADOOP_CONF_DIR* is the is the directory for storing the Hadoop configuration files. For example, `/etc/hadoop/conf`.

c. Copy the Ganglia metrics properties file into place.

```
mv hadoop-metrics2.properties-GANGLIA  hadoop-metrics2.properties
```

d. Edit the metrics properties file and set the Ganglia server hostname.

```
namenode.sink.ganglia.servers=$my.ganglia.server.hostname:8661
datanode.sink.ganglia.servers=$my.ganglia.server.hostname:8660
jobtracker.sink.ganglia.servers=$my.ganglia.server.hostname:8662
tasktracker.sink.ganglia.servers=$my.ganglia.server.hostname:8660
maptask.sink.ganglia.servers=$my.ganglia.server.hostname:8660
reducetask.sink.ganglia.servers=$my.ganglia.server.hostname:8660
```

e. Restart the Hadoop services using the instructions provided here.

11.3. Validate the Installation

Use the following instructions to validate your installation:

1. Start the Ganglia Server. Execute the following command on the Ganglia server:

```
service httpd restart
/etc/init.d/hdp-gmetad start
```

2. Start Ganglia monitoring on all hosts. Execute the following command on all hosts:

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

3. Confirm that Ganglia is Running. Browse to the Ganglia server:

```
http://$my.ganglia.server.hostname/ganglia
```

12. Installing Nagios

This section describes installing and testing Nagios, a system that monitors Hadoop cluster components and issues alerts on warning and critical conditions.

Use the following instructions to install Nagios:

1. Install the Nagios RPMs
2. Install the configuration files
3. Validate the installation

12.1. Install the Nagios RPMs

On the host you have chosen to be the Nagios server, install the RPMs:

- For RHEL and CentOS

```
yum install net-snmp net-snmp-utils php-pecl-json
yum install wget httpd php net-snmp-perl perl-Net-SNMP fping nagios-3.2.3
 nagios-plugins-1.4.9 hdp_mon_nagios_addons
```

- For SLES

```
zypper install net-snmp
zypper install wget apache2 php php-curl perl-SNMP perl-Net-SNMP fping
 nagios-3.2.3-2.1 nagios-plugins-1.4.9 hdp_mon_nagios_addons
```

12.2. Install the Configuration Files

There are several configuration files that need to be set up for Nagios. Use the following instructions to install and setup the configuration files for Nagios:

1. Extract the Nagios configuration files.

   From the `scripts.zip` file (downloaded in Download Companion Files), copy the files from `configuration_files/nagios` directory to a temporary directory.

   The `nagios` directory contains two sub-directories, `objects` and `plugins`.

2. Copy the configuration files.

   a. Copy the contents of the `objects` directory to `/etc/nagios/objects` :

   ```
   cp $tmp-directory/nagios/objects/* /etc/nagios/objects/*
   ```

   b. Copy the contents of the `plugins` directory to the following location:

   ```
   cp $tmp-directory/nagios/plugins/* /usr/lib64/nagios/plugins/
   ```

3. Set the Nagios Admin password.

   a. Choose a Nagios administrator password, for example, `admin`.

   b. Use the following command to set the password:

```
htpasswd -c -b  /etc/nagios/htpasswd.users nagiosadmin admin
```

4. Set the Nagios Admin email contact address. Edit the `/etc/nagios/objects/contacts.cfg` file and change the `nagios@localhost` value to the admin email address for receiving alerts.

5. Register the Hadoop configuration files:

   Edit the `/etc/nagios/nagios.cfg` file to add the following values in the `OBJECT CONFIGURATION FILE(S)` section:

```
# Definitions for hadoop servers
cfg_file=/etc/nagios/objects/hadoop-commands.cfg
cfg_file=/etc/nagios/objects/hadoop-hosts.cfg
cfg_file=/etc/nagios/objects/hadoop-hostgroups.cfg
cfg_file=/etc/nagios/objects/hadoop-services.cfg
cfg_file=/etc/nagios/objects/hadoop-servicegroups.cfg
```

6. Set Hosts.

   a. Edit the `/etc/nagios/objects/hadoop-hosts.cfg` file and add a "`define host { … }`" entry for each host in your cluster using the following format:

```
define host {
       alias @HOST@
       host_name @HOST@
       use linux-server
       address @HOST@
       check_interval 0.25
       retry_interval 0.25
       max_check_attempts 4
       notifications_enabled 1
       first_notification_delay 0 # Send notification soon after
                                  change in the hard state
       notification_interval 0    # Send the notification once
       notification_options       d,u,r
  }
```

   b. Replace the "@HOST@" with the hostname.

7. Set Host Groups

   a. Open `/etc/nagios/objects/hadoop-hostsgroups.cfg` with a text editor.

   b. Create host groups based on all the hosts and services you have installed in your cluster. Each host group entry should follow this format:

```
define hostgroup {
       hostgroup_name  @NAME@
       alias           @ALIAS@
       members         @MEMBERS@
   }
```

   Where

Table 12.1. Host Group Parameters

Parameter	Description
@NAME@	The host group name
@ALIAS@	The host group alias
@MEMBERS@	A comma-separated list of hosts in the group

c. The following table lists the core and monitoring host groups:

Table 12.2. Core and Monitoring Hosts

Service	Component	Name	Alias	Members
All servers in the cluster		all-servers	All Servers	List all servers in the cluster
HDFS	NameNode	namenode	namenode	The NameNode host
HDFS	SecondaryNameNode	snamenode	snamenode	The Secondary NameNode host
MapReduce	JobTracker	jobtracker	jobtracker	The Job Tracker host
HDFS, MapReduce	Slaves	slaves	slaves	List all hosts running DataNode and TaskTrackers
Nagios		nagios-server	nagios-server	The Nagios server host
Ganglia		ganglia-server	ganglia-server	The Ganglia server host

d. The following table lists the ecosystem project host groups:

Table 12.3. Ecosystem Hosts

Service	Component	Name	Alias	Members
HBase	Master	hbasemaster	hbasemaster	List the master server
HBase	Region	regions-servers	region-servers	List all region servers
ZooKeeper		zookeeper-servers	zookeeper-servers	List all ZooKeeper servers
Oozie		oozie-server	oozie-server	The Oozie server
Hive		hiveserver	hiverserver	The Hive metastore server
WebHCat		twebhcat-server	webhcat-server	The WebHCat server

8. Set Services.

a. Open `/etc/nagios/objects/hadoop-services.cfg` with a text editor.

This file contains service definitions for the following services: Ganglia, HBase (Master and Region), ZooKeeper, Hive, WebHCat, and Oozie

b. Remove any services definitions for services you have not installed.

c. Replace the `@NAGIOS_BIN@` and `@STATUS_DAT@` parameters as shown below:

- For RHEL and CentOS

```
@STATUS_DAT@ = /var/nagios/status.dat
@NAGIOS_BIN@ = /usr/bin/nagios
```

- For SLES

```
@STATUS_DAT@ = /var/lib/nagios/status.dat
@NAGIOS_BIN@ = /usr/sbin/nagios
```

d. If you have installed Hive or Oozie services, replace the parameter `@JAVA_HOME@` with the path to the Java home. For example, `/usr/java/default.`

9. Set Status.

a. Open `/etc/nagios/objects/hadoop-commands.cfg` with a text editor.

b. Replace the @STATUS_DAT@ parameter with the location of the Nagios status file as shown below:

- For RHEL and CentOS

```
/var/nagios/status.dat
```

- For SLES

```
 /var/lib/nagios/status.dat
```

12.3. Validate the Installation

Use the following instructions to validate your installation:

1. Start the Nagios server. Execute the following command to start the Nagios server:

```
/etc/init.d/nagios start
```

2. Confirm the server is running

```
/etc/init.d/nagios status
```

This should return:

```
nagios (pid #) is running...
```

3. Test Nagios Services. On the Nagios host machine, execute the following command:

```
/usr/lib64/nagios/plugins/check_hdfs_capacity.php -h namenode_hostname -p
 50070 -w 80% -c 90%
```

This should return:

```
OK: DFSUsedGB:<some#>, DFSTotalGB:<some#>
```

4. Test Nagios Access.

a. Browse to the Nagios server:

```
http://$nagios.server/nagios
```

b. Login using the Nagios admin username (nagiosadmin) and password.

c. Click on **hosts** to validate that all the hosts in the cluster are listed.

d. Click on **services** to validate all the Hadoop services are listed for each host.

5. Test Nagios Alerts.

a. Login to one of your cluster DataNodes.

b. Stop the TaskTracker service.

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

c. Validate that you received an alert at the admin email address and that you have critical state showing on the console.

d. Start the TaskTracker service.

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

e. Validate that you received an alert at the admin email address and that critical state is cleared on the console.

13. Setting Up Security for Manual Installs

This section provides information on enabling security for a manually installed version of HDP. Use the following instructions to deploy secure Hadoop cluster:

1. Preparing Kerberos
2. Configuring HDP

13.1. Preparing Kerberos

This section provides information on setting up Kerberos for an HDP installation.

1. Kerberos Overview
2. Installing and Configuring the KDC
3. Creating the Database and Setting Up the First Administrator
4. Creating Service Principals and Keytab Files for HDP
5. Providing the jce-6 Security JAR files

13.1.1. Kerberos Overview

To create secure communication among its various components, HDP uses Kerberos. Kerberos is a third party authentication mechanism, in which users and services that users wish to access rely on a third party - the Kerberos server - to authenticate each to the other. This mechanism also supports encrypting all traffic between the user and the service. The Kerberos server itself is known as the *Key Distribution Center*, or KDC. At a high level, it has three parts:

- A database of the users and services (known as *principals*) that it knows about and their respective Kerberos passwords
- An *authentication server* (AS) which performs the initial authentication and issues a *Ticket Granting Ticket* (TGT)
- A *Ticket Granting Server* (TGS) that issues subsequent service tickets based on the initial TGT.

A user principal requests authentication from the AS. The AS returns a TGT that is encrypted using the user principal's Kerberos password, which is known only to the user principal and the AS. The user principal decrypts the TGT locally using its Kerberos password, and from that point forward, until the ticket expires, the user principal can use the TGT to get service tickets from the TGS.

Because a service principal cannot provide a password each time to decrypt the TGT, it uses a special file, called a *keytab,* which contains its authentication credentials.

The service tickets are what allow the principal to access various services. The set of hosts, users, and services over which the Kerberos server has control is called a *realm*.

Note

Because Kerberos is a time-sensitive protocol, all hosts in the realm must be time-synchronized, for example, by using the Network Time Protocol (NTP). If the local system time of a client differs from that of the KDC by as little as 5 minutes (the default), the client will not be able to authenticate.

13.1.2. Installing and Configuring the KDC

To use Kerberos with HDP you can either use an existing KDC or install a new one just for HDP's use. The following gives a very high level description of the installation process. To get more information see RHEL documentation or CentOS documentation or SLES documentation.

To install a new version of the server:

```
[On RHEL or CentOS]
yum install krb5-server krb5-libs krb5-auth-dialog krb5-workstation
```

OR

```
[On SLES]
zypper install krb5 krb5-server krb5-client
```

Note

The host on which you install the KDC must itself be secure.

When the server is installed you must edit the two main configuration files, located by default here:

[On RHEL or CentOS]

- `/etc/krb5.conf`
- `/var/kerberos/krb5kdc/kdc.conf`.

OR

[On SLES]

- `/etc/krb5.conf`
- `/var/lib/kerberos/krb5kdc/kdc.conf`

Use these files to specify the realm by changing EXAMPLE.COM and example.com to case-matched version of the domain name for the realm and changing the KDC value from `kerberos.example.com` to the fully qualified name of the Kerberos server host.

The updated version of `/etc/krb5.conf` should be copied to every node in your cluster.

13.1.3. Creating the Database and Setting Up the First Administrator

1. Use the utility `kdb5_util` to create the Kerberos database.

   ```
    [on RHEL or CentOS]
   /usr/sbin/kdb5_util create -s
   ```

   OR

   ```
   [on SLES]
    kdb5_util create -s
   ```

   The -s option allows you to store the master server key for the database in a *stash* file. If the stash file is not present, you will need to log into the KDC with the master password (specified during installation) each time it starts. This will automatically regenerate the master server key.

2. Edit the Access Control List (`/var/kerberos/krb5kdc/kadm5.acl` in RHEL or CentOS and `/var/lib/kerberos/krb5kdc/kadm5.acl` in SLES) to define the principals that have admin (modifying) access to the database. A simple example would be a single entry:

   ```
   */admin@EXAMPLE.COM *
   ```

   This specifies that all principals with the `/admin` *instance* extension have full access to the database. You must restart `kadmin` for the change to take effect.

3. Create the first user principal. This must be done at a terminal window on the KDC machine itself, while you are logged in as `root`. Notice the `.local`. Normal `kadmin` usage requires that a principal with appropriate access already exist.The `kadmin.local` command can be used even if no principals exist.

   ```
   /usr/sbin/kadmin.local -q "addprinc <username>/admin"
   ```

   Other principals can now be created either on the KDC machine itself or through the network, using this principal. The following instruction assume you are using the KDC machine.

4. Start Kerberos.

   ```
   [on RHEL and CentOS]
   /sbin/service krb5kdc start
   /sbin/service kadmin start
   ```

   OR

   ```
   [on SLES]
   rckrb5kdc start
   rckadmind start
   ```

13.1.4. Creating Service Principals and Keytab Files for HDP

Each service in HDP must have its own principal. As services do not login with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file,

which is extracted from the Kerberos database and stored locally with the service principal. First you must create the principal, using mandatory naming conventions. Then you must create the keytab file with that principal's information and copy the file to the keytab directory on the appropriate service host.

1. Create a service principal using the `kadmin` utility:

   ```
   kadmin: addprinc -randkey $principal_name/$fully.qualified.domain.name@YOUR-REALM.COM
   ```

   You must have a principal with administrative permissions to use this command. The randkey is used to generate the password.

   Note that in the example each service principal's name has appended to it the fully qualified domain name of the host on which it is running. This is to provide a unique principal name for services that run on multiple hosts, like DataNodes and TaskTrackers.

   The addition of the hostname serves to distinguish, for example, a request from DataNode A from a request from DataNode B. This is important for two reasons:

   - If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised
   - If multiple DataNodes have exactly the same principal and are simultaneously connecting to the NameNode, and if the Kerberos authenticator being sent happens to have same timestamp, then the authentication would be rejected as a replay request.

   The *`$principal_name`* part of the name must match the values in the table below:

   Note that the NameNode, Secondary NameNode, and Oozie require two principals each.

   Table 13.1. Service Principal Names

Service Name	Mandatory Principal Name
NameNode	`nn` AND HTTP
Secondary NameNode	`nn` AND HTTP
JobTracker	`jt`
TaskTracker	`tt`
DataNode	`dn`
HBase Master	`hbase`
HBase RegionServer	`hbase`
ZooKeeper	`zookeeper`
HCatalog Server	`hcat`
Oozie	`oozie` and HTTP
WebHCat (Templeton)	HTTP

   For example: To create the principal for a DataNode service, issue this command:

   ```
   kadmin: addprinc -randkey dn/$DataNode-Host@EXAMPLE.COM
   ```

2. Extract the related keytab file and place it in the keytab directory (by default `/etc/krb5.keytab`) of the appropriate respective components:

```
kadmin: xst -norandkey -k $keytab_file_name $principal_name/fully.qualified.
domain.name
```

You must use the mandatory names for the *$keytab_file_name*; variable shown in this table.

Table 13.2. Service Keytab File Names

Service Name	Mandatory Keytab File Name

NameNode	`nn.service.keytab`
	AND
	`spnego.service.keytab`

Secondary NameNode	`nn.service.keytab`
	AND
	`spnego.service.keytab`

JobTracker	`jt.service.keytab`
TaskTracker	`tt.service.keytab`
DataNode	`dn.service.keytab`
HBase Master	`hbase.service.keytab`
HBase RegionServer	`hbase.service.keytab`
ZooKeeper	`zookeeper.service.keytab`
HCatalog Server	`hcat.service.keytab`

Oozie	`oozie.service.keytab`
	AND
	`spnego.service.keytab`

Templeton	`spnego.service.keytab`

For example: To create the keytab files for the NameNode, issue these commands:

```
 kadmin: xst -k nn.service.keytab nn/<namenode-host>
kadmin: xst -k spnego.service.keytab HTTP/<namenode-host>
```

When you have created the keytab files, copy them to the `keytab` directory of the respective service hosts.

3. Set appropriate permissions for the keytabs.

   a. Secure all the keytabs. Execute the following command on all the hosts on your cluster:

   ```
   chown -R root:hadoop /etc/security/keytabs
   chmod -R g+rX,o= /etc/security/keytabs
   ```

   b. On the NameNode, execute the following command:

   ```
   chown hdfs:hadoop /etc/security/keytabs/nn.service.keytab
   chmod 400 /etc/security/keytabs/nn.service.keytab
   ```

   c. Execute the following command on all the slave nodes:

   ```
   chown hdfs:hadoop /etc/security/keytabs/dn.service.keytab
   ```

```
chown mapred:hadoop /etc/security/keytabs/tt.service.keytab
chmod 400 /etc/security/keytabs/*.service.keytab
```

4. Verify that the correct keytab files and principals are associated with the correct service using the `klist` command. For example, on the NameNode:

```
klist –k -t /etc/security/nn.service.keytab
```

   Do this on each respective service in your cluster.

13.1.5. Providing the jce-6 Security JAR files

You must have a copy of the jce-6 security policy jars available in the `$JAVA_HOME/jre/lib/security/` directory of each of your hosts. You can download them from here:

```
http://www.oracle.com/technetwork/java/javase/downloads/jce-6-download-429243.html
```

13.2. Configuring HDP

This section provides information on configuring HDP for Kerberos.

- Configuration Overview
- Creating Mappings Between Principals and UNIX Usernames
- Adding Security Information to Configuration Files

13.2.1. Configuration Overview

Configuring HDP for Kerberos has two parts:

- Creating a mapping between service principals and UNIX usernames.

  Hadoop uses group memberships of users at various places, such as to determine group ownership for files or for access control.

  A user is mapped to the groups it belongs to using an implementation of the `GroupMappingServiceProvider` interface. The implementation is pluggable and is configured in `core-site.xml`.

  By default Hadoop uses `ShellBasedUnixGroupsMapping`, which is an implementation of `GroupMappingServiceProvider`. It fetches the group membership for a username by executing a UNIX shell command. In secure clusters, since the usernames are actually Kerberos principals, `ShellBasedUnixGroupsMapping` will work only if the Kerberos principals map to valid UNIX usernames. Hadoop provides a feature that lets administrators specify mapping rules to map a Kerberos principal to a local UNIX username .

- Adding information to three main service configuration files.

  There are several optional entries in the three main service configuration files that must be added to enable security on HDP.

13.2.2. Creating Mappings Between Principals and UNIX Usernames

HDP uses a rule-based system to create mappings between service principals and their related UNIX usernames. The rules are specified in the `core-site.xml` configuration file as the value to the optional key `hadoop.security.auth_to_local`.

The default rule is simply named `DEFAULT`. It translates all principals in your default domain to their first component. For example, `myusername@APACHE.ORG` and `myusername/admin@APACHE.ORG` both become `myusername`, assuming your default domain is APACHE.ORG.

Use the following instructions to configure the mappings between principals and UNIX usernames:

1. Creating Rules
2. Examples

13.2.2.1. Creating Rules

To accomodate more complex translations, you can create a hierarchical set of rules to add to the default. Each rule is divided into three parts: base, filter, and substitution.

- **The Base:**

  The base begins with the number of components in the principal name (excluding the realm), followed by a colon, and the pattern for building the username from the sections of the principal name. In the pattern section `$0` translates to the realm, `$1` translates to the first component and `$2` to the second component.

  For example:

  `[1:$1@$0]` translates `myusername@APACHE.ORG` to `myusername@APACHE.ORG`

  `[2:$1]` translates `myusername/admin@APACHE.ORG` to `myusername`

  `[2:$1%$2]` translates `myusername/admin@APACHE.ORG` to “`myusername%admin`

- **The Filter:**

  The filter consists of a regex in a parentheses that must match the generated string for the rule to apply.

  For example:

  `(.*%admin)` matches any string that ends in `%admin`

  `(.*@SOME.DOMAIN)` matches any string that ends in `@SOME.DOMAIN`

- **The Substitution:**

  The substitution is a sed rule that translates a regex into a fixed string.

For example:

`s/@ACME\.COM//` removes the first instance of `@SOME.DOMAIN`.

`s/@[A-Z]*\.COM//` removes the first instance of `@` followed by a name followed by `COM`.

`s/X/Y/g` replaces all of the `X` in the name with `Y`

13.2.2.2. Examples

- If your default realm was `APACHE.ORG`, but you also wanted to take all principals from `ACME.COM` that had a single component `joe@ACME.COM`, you would create this rule:

```
RULE:[1:$1@$0](.@ACME.COM)s/@.//
DEFAULT
```

- To also translate names with a second component, you would use these rules:

```
RULE:[1:$1@$0](.@ACME.COM)s/@.//
RULE:[2:$1@$0](.@ACME.COM)s/@.//
DEFAULT
```

- To treat all principals from `APACHE.ORG` with the extension `/admin` as `admin`, your rules would look like this:

```
RULE[2:$1%$2@$0](.%admin@APACHE.ORG)s/./admin/
DEFAULT
```

13.2.3. Adding Security Information to Configuration Files

To enable security on HDP, you must add optional information to various configuration files. Use the following instructions to configure security information:

1. Configure Secure Hadoop
2. Configure Secure HBase and ZooKeeper
3. Configure Secure Hive
4. Configure Secure Oozie
5. Configure Secure WebHCat

13.2.3.1. Configure Secure Hadoop

1. Edit the `core-site.xml` file on *every* host in your cluster, to add the following information:

```
<property>
       <name>hadoop.security.authentication</name>
       <value>kerberos</value>
       <description>Set the authentication for the cluster. Valid values
 are: simple or kerberos.
       </description>
</property>
```

```
<property>
        <name>hadoop.rpc.protection</name>
        <value>authentication</value>
        <description>This is an [OPTIONAL] setting. If not set, defaults
 to authentication.authentication= authentication only; the client
 and server mutually authenticate during connection setup.integrity
 = authentication and integrity; guarantees the integrity of data
 exchanged between client and server aswell as authentication.privacy
 = authentication, integrity, and confidentiality; guarantees that data
 exchanged between client andserver is encrypted and is not readable by a
 “man in the middle”.
        </description>
</property>
```

```
<property>
        <name>hadoop.security.authorization</name>
        <value>true</value>
        <description>Enable authorization for different protocols.
        </description>
</property>
```

```
<property>

        <name>hadoop.security.auth_to_local</name>
        <value>RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/$MAPRED_USER/
RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/$HDFS_USER/
DEFAULT</value>
        <description>The mapping from Kerberos principal names to local OS
 user names. </description>
</property>
```

For mapping from Kerberos principal names to local OS user names, see Creating Mappings Between Principals and UNIX Usernames.

```
<property>
  <name>hadoop.proxyuser.hive.groups</name>
  <value>users</value>
  <description>Allow the superuser hive to impersonate any members of the
 group users. Required only when installing Hive.
  </description>
</property>
```

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

```
<property>
  <name>hadoop.proxyuser.hive.hosts</name>
  <value>$Hive_Hostname_FQDN</value>
  <description>Hostname from where superuser hive can connect. Required only
 when installing Hive.
  </description>
</property>
```

```
<property>
  <name>hadoop.proxyuser.oozie.groups</name>
 <value>users</value>
  <description>Allow the superuser oozie to impersonate any members of the
 group users. Required only when installing Oozie.
 </description>
</property>
```

```
<property>
  <name>hadoop.proxyuser.oozie.hosts</name>
  <value>$Oozie_Hostname_FQDN</value>
 <description>Hostname from where superuser oozie can connect. Required only
 when installing Oozie.
  </description>
</property>
```

```
<property>
 <name>hadoop.proxyuser.hcat.hosts</name>
  <value>$WebHCat_Hostname_FQDN</value>
  <description>Hostname from where superuser hcat can connect. Required only
 when installing WebHCat.
  </description>
</property>
```

```
<property>
  <name>hadoop.proxyuser.HTTP.groups</name>
 <value>users</value>
 <description>Allow the superuser HTTP to impersonate any members of the
 group users.
  </description>
</property>
```

```
<property>
  <name>hadoop.proxyuser.HTTP.hosts</name>
  <value>$WebHCat_Hostname_FQDN</value>
 <description>Hostname from where superuser HTTP can connect.
 </description>
</property>
```

```
 <property>
  <name>hadoop.proxyuser.hcat.groups</name>
 <value>users</value>
  <description>Allow the superuser hcat to impersonate any members of the
 group users. Required only when installing WebHCat.
 </description>
</property>
```

```
<property>
 <name>hadoop.proxyuser.hcat.hosts</name>
  <value>$WebHCat_Hostname_FQDN</value>
  <description>Hostname from where superuser hcat can connect. This is
 required only when installing webhcat on the cluster.
  </description>
</property>
```

2. Edit the `hdfs-site.xml` file on *every* host in your cluster, to add the following information:

```
<property>
        <name>dfs.block.access.token.enable</name>
        <value>true</value>
        <description> If "true", access tokens are used as capabilities
        for accessing datanodes. If "false", no access tokens are checked on
        accessing datanodes. </description>
</property>
```

```
<property>
        <name>dfs.namenode.kerberos.principal</name>
        <value>nn/_HOST@EXAMPLE.COM</value>
        <description> Kerberos principal name for the
        NameNode </description>
</property>

<property>
        <name>dfs.secondary.namenode.kerberos.principal</name>
        <value>nn/_HOST@EXAMPLE.COM</value>
        <description>Kerberos principal name for the secondary NameNode.
        </description>
</property>

<property>
        <!--cluster variant -->
        <name>dfs.secondary.http.address</name>
        <value>$Secondary.NameNode.FQDN</value>
        <description>Address of secondary namenode web server</description>

</property>

<property>
        <name>dfs.secondary.https.port</name>
        <value>50490</value>
        <description>The https port where secondary-namenode
        binds</description>
</property>

<property>
        <name>dfs.web.authentication.kerberos.principal</name>
        <value>HTTP/_HOST@EXAMPLE.COM</value>
        <description> The HTTP Kerberos principal used by Hadoop-Auth in the
 HTTP endpoint.
The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP SPNEGO
 specification.
        </description>
</property>

<property>
        <name>dfs.web.authentication.kerberos.keytab</name>
        <value>/etc/security/keytabs/spnego.service.keytab</value>
        <description>The Kerberos keytab file with the credentials for the
 HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
        </description>
</property>

<property>
        <name>dfs.datanode.kerberos.principal</name>
        <value>dn/_HOST@EXAMPLE.COM</value>
        <description>The Kerberos principal that the DataNode runs as.
 "_HOST" is replaced by the real host name.
        </description>
</property>
```

```
<property>
        <name>dfs.namenode.keytab.file</name>
        <value>/etc/security/keytabs/nn.service.keytab</value>
        <description>Combined keytab file containing the NameNode service
 and host principals.
        </description>
</property>

<property>
        <name>dfs.secondary.namenode.keytab.file</name>
        <value>/etc/security/keytabs/nn.service.keytab</value>
        <description>Combined keytab file containing the NameNode service
 and host principals.
        </description>
</property>

<property>
        <name>dfs.datanode.keytab.file</name>
        <value>/etc/security/keytabs/dn.service.keytab</value>
        <description>The filename of the keytab file for the DataNode.
        </description>
</property>

<property>
        <name>dfs.https.port</name>
        <value>50470</value>
        <description>The https port where NameNode binds</description>
</property>

<property>
        <name>dfs.https.address</name>
        <value>$HTTPS_Address_for_NameNode</value>
        <description>The https address where namenode binds. Example:
 ip-10-111-59-170.ec2.internal:50470</description>
</property>

<property>
        <name>dfs.namenode.kerberos.internal.spnego.principal</name>
        <value>$dfs.web.authentication.kerberos.principal</value>
</property>

<property>
        <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</
name>
        <value>$dfs.web.authentication.kerberos.principal</value>
</property>

<property>
        <name>dfs.datanode.address</name>
        <value></value>
        <description>The address, with a privileged port - any port number
 under 1023. Example: 0.0.0.0:1019</description>
</property>

<property>
        <name>dfs.datanode.http.address</name>
        <value>The address, with a privileged port - any port number under
 1023. Example: 0.0.0.0:1022</value>
</property>
```

For the DataNodes to run in secure mode, you must set the user-name which the DataNode process should run as by setting HADOOP_SECURE_DN_USER as shown below:

```
export HADOOP_SECURE_DN_USER=$HDFS_USER
```

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

Note

The DataNode daemon must be started as `root`.

Optionally, you can allow that user to access the directories where PID and log files are stored. For example:

```
export HADOOP_SECURE_DN_PID_DIR=/var/run/hadoop/$HADOOP_SECURE_DN_USER
export HADOOP_SECURE_DN_LOG_DIR=/var/run/hadoop/$HADOOP_SECURE_DN_USER
```

3. Edit the `mapred-site.xml` file on *every* host in your cluster to add the following information:

```
<property>
        <name>mapreduce.jobtracker.kerberos.principal</name>
        <value>jt/_HOST@EXAMPLE.COM</value>
        <description>Kerberos principal name for the JobTracker   </
description>
</property>
```

```
<property>
        <name>mapreduce.tasktracker.kerberos.principal</name>
        <value>tt/_HOST@EXAMPLE.COM</value>
        <description>Kerberos principal name for the TaskTracker."_HOST" is
 replaced by the host name of the TaskTracker.
        </description>
</property>
```

```
<property>
        <name>mapreduce.jobtracker.keytab.file</name>
        <value>/etc/security/keytabs/jt.service.keytab</value>
        <description>The keytab for the JobTracker principal.
        </description>
</property>
```

```
<property>
        <name>mapreduce.tasktracker.keytab.file</name>
        <value>/etc/security/keytabs/tt.service.keytab</value>
        <description>The filename of the keytab for the TaskTracker</
description>
</property>
```

```
<property>
        <name>mapreduce.jobhistory.kerberos.principal</name>
        <!--cluster variant -->
        <value>jt/_HOST@EXAMPLE.COM</value>
        <description> Kerberos principal name for JobHistory. This must map
 to the same user as the JobTracker user (mapred).
        </description>
</property>
```

```
<property>
        <name>mapreduce.jobhistory.keytab.file</name>
        <!--cluster variant -->
        <value>/etc/security/keytabs/jt.service.keytab</value>
        <description>The keytab for the JobHistory principal.
        </description>
</property>
```

13.2.3.2. Configure Secure HBase and ZooKeeper

Use the following instructions to set up secure HBase and ZooKeeper:

1. Configure HBase Master
2. Create JAAS Configuration Files
3. Start HBase and ZooKeeper Services
4. Configure Secure Client Side Access HBase
5. Optional: Configure Client-Side Operation For Secure Operation - Thrift Gateway
6. Optional: Configure Client-Side Operation For Secure Operation - REST Gateway
7. Configure HBase for Access Control Lists (ACL)

13.2.3.2.1. Configure HBase Master

Edit `hbase-site.xml` file on your HBase Master server to add the following information:

Note

There are no default values; the following are all only examples.

```
  <property>
        <name>hbase.master.keytab.file</name>
        <value>/etc/security/keytabs/hbase.service.keytab</value>
        <description>Full path to the kerberos keytab file to use
                     for logging in the configured HMaster server principal.

        </description>
</property>
```

```
<property>
        <name>hbase.master.kerberos.principal</name>
        <value>hbase/_HOST@EXAMPLE.COM</value>
        <description>Ex. "hbase/_HOST@EXAMPLE.COM".
        The kerberos principal name that should be used to run the HMaster
 process.
        The principal name should be in the form: user/hostname@DOMAIN.  If
 "_HOST" is used as the hostname portion,
        it will be replaced with the actual hostname of the running instance.

        </description>
</property>
```

```
<property>
        <name>hbase.regionserver.keytab.file</name>
        <value>/etc/security/keytabs/hbase.service.keytab</value>
        <description>Full path to the kerberos keytab file to use for logging
        in the configured HRegionServer server principal.
        </description>
</property>
```

```
<property>
        <name>hbase.regionserver.kerberos.principal</name>
        <value>hbase/_HOST@EXAMPLE.COM</value>
        <description>Ex. "hbase/_HOST@EXAMPLE.COM".The kerberos principal name
 thatshould be used to run the HRegionServer process.
The principal name should be in the form: user/hostname@DOMAIN.
If _HOSTis used as the hostname portion, it will be replaced with the actual
 hostname of the runninginstance.
An entry for this principal must existin the file specified in hbase.
regionserver.keytab.file
        </description>
</property>
```

```
<!--Additional configuration specific to HBase security -->

<property>
        <name>hbase.superuser</name>
        <value>hbase</value>
        <description>List of users or groups (comma-separated), who are
 allowed full privileges, regardless of stored ACLs, across the cluster.
Only used when HBase security is enabled.
        </description>
</property>
```

```

<property>
        <name>hbase.coprocessor.region.classes</name>
        <value>org.apache.hadoop.hbase.security.token.TokenProvider,org.
apache.hadoop.hbase.security.access.SecureBulkLoadEndpoint,org.apache.hadoop.
hbase.security.access.AccessController </value>
        <description>A comma-separated list of Coprocessors that are loaded by
 default on all tables.
        </description>
</property>
```

```
<property>
        <name>hbase.security.authentication</name>
        <value>kerberos</value>

</property>
```

```
<property>
        <name>hbase.rpc.engine</name>
        <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>

</property>
```

```
<property>
        <name>hbase.security.authorization</name>
        <value>true</value>
        <description>Enables HBase authorization. Set the value of this
 property to false to disable HBase authorization.
        </description>

</property>
```

```
<property>
        <name>hbase.coprocessor.master.classes</name>
        <value>org.apache.hadoop.hbase.security.access.AccessController</
value>

</property>
```

```
<property>
        <name>hbase.bulkload.staging.dir</name>
        <value>/apps/hbase/staging</value>
        <description>Directory in the default filesystem, owned by the hbase
 user, and has permissions(-rwx--x--x, 711) </description>

</property>
```

For more information on bulk loading in secure mode, see HBase Secure BulkLoad. Note that the `hbase.bulkload.staging.dir` is created by HBase.

13.2.3.2.2. Create JAAS Configuration Files

1. Create the following JAAS configuration files on the HBase Master, RegionServer, and HBase client host machines.

   These files must be created under the *$HBASE_CONF_DIR* directory:

   where *$HBASE_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`.

   - On your HBase Master host machine, create the `hbase-server.jaas` file under the `/etc/hbase/conf` directory and add the following content:

     ```
     Server {
     com.sun.security.auth.module.Krb5LoginModule required
     useKeyTab=true
     storeKey=true
     useTicketCache=false
     keyTab="/etc/security/keytabs/hbase.service.keytab"
     principal="hbase/$HBase.Master.hostname";
     };
     ```

   - On each of your RegionServer host machine, create the `regionserver.jaas` file under the `/etc/hbase/conf` directory and add the following content:

     ```
     Server {
     com.sun.security.auth.module.Krb5LoginModule required
     useKeyTab=true
     storeKey=true
     useTicketCache=false
     keyTab="/etc/security/keytabs/hbase.service.keytab"
     principal="hbase/$RegionServer.hostname";
     };
     ```

- On HBase client machines, create the `hbase-client.jaas` file under the `/etc/hbase/conf` directory and add the following content:

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=false
useTicketCache=true;
};
```

2. Create the following JAAS configuration files on the ZooKeeper Server and client host machines.

   These files must be created under the *$ZOOKEEPER_CONF_DIR* directory:

   where *$ZOOKEEPER_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/zookeeper/conf`.

   - On ZooKeeper server host machines, create the `zookeeper-server.jaas` file under the `/etc/zookeeper/conf` directory and add the following content:

```
Server {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
storeKey=true
useTicketCache=false
keyTab="/etc/security/keytabs/zookeeper.service.keytab"
principal="zookeeper/$ZooKeeper.Server.hostname";
};
```

   - On ZooKeeper client host machines, create the `zookeeper-client.jaas` file under the `/etc/zookeeper/conf` directory and add the following content:

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=false
useTicketCache=true;
};
```

3. Edit the `hbase-env.sh` file on your HBase server to add the following information:

```
export HBASE_OPTS ="-Djava.security.auth.login.config=$HBASE_CONF_DIR/hbase-
client.jaas"
export HBASE_MASTER_OPTS ="-Djava.security.auth.login.config=
$HBASE_CONF_DIR/hbase-server.jaas"
export HBASE_REGIONSERVER_OPTS="-Djava.security.auth.login.config=
$HBASE_CONF_DIR/regionserver.jaas"
```

   where *HBASE_CONF_DIR* is the HBase configuration directory. For example, `/etc/hbase/conf`.

4. Edit `zoo.cfg` file on your ZooKeeper server to add the following information:

```
authProvider.1=org.apache.zookeeper.server.auth.SASLAuthenticationProvider
jaasLoginRenew=3600000
kerberos.removeHostFromPrincipal=true
kerberos.removeRealmFromPrincipal=true
```

5. Edit `zookeeper-env.sh` file on your ZooKeeper server to add the following information:

```
export SERVER_JVMFLAGS ="-Djava.security.auth.login.
config=$ZOOKEEPER_CONF_DIR/zookeeper-server.jaas"
export CLIENT_JVMFLAGS ="-Djava.security.auth.login.
config=$ZOOKEEPER_CONF_DIR/zookeeper-client.jaas"
```

   where *$ZOOKEEPER_CONF_DIR* is the ZooKeeper configuration directory. For example, `/etc/zookeeper/conf`.

13.2.3.2.3. Start HBase and ZooKeeper Services

Start the HBase and ZooKeeper services using the instructions provided here.

If the configuration is successful, you should see the following in your ZooKeeper server logs:

```
11/12/05 22:43:39 INFO zookeeper.Login: successfully logged in.
11/12/05 22:43:39 INFO server.NIOServerCnxnFactory: binding to port 0.0.0.0/0.
0.0.0:2181
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh thread started.
11/12/05 22:43:39 INFO zookeeper.Login: TGT valid starting at:        Mon Dec
 05 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT expires:                  Tue Dec
 06 22:43:39 UTC 2011
11/12/05 22:43:39 INFO zookeeper.Login: TGT refresh sleeping until: Tue Dec 06
 18:36:42 UTC 2011
..
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler:
  Successfully authenticated client: authenticationID=hbase/ip-10-166-175-249.
us-west-1.compute.internal@HADOOP.LOCALDOMAIN;
  authorizationID=hbase/ip-10-166-175-249.us-west-1.compute.internal@HADOOP.
LOCALDOMAIN.
11/12/05 22:43:59 INFO auth.SaslServerCallbackHandler: Setting authorizedID:
 hbase
11/12/05 22:43:59 INFO server.ZooKeeperServer: adding SASL authorization for
 authorizationID: hbase
```

13.2.3.2.4. Configure Secure Client Side Access HBase

HBase configured for secure client access is expected to be running on top of a secure HDFS cluster. HBase must be able to authenticate to HDFS services.

1. Provide a Kerberos principal to the HBase client user using the instructions provided here.

   - Option I: Provide Kerberos principal to normal HBase clients.

     For normal HBase clients, Hortonworks recommends setting up a password to the principal.

   - Set `maxrenewlife`.

     The client principal's `maxrenewlife` should be set high enough so that it allows enough time for the HBase client process to complete. Client principals are not renewed automatically.

For example, if a user runs a long-running HBase client process that takes at most three days, we might create this user's principal within kadmin with the following command:

```
addprinc -maxrenewlife 3days
```

- Option II: Provide Kerberos principal to long running HBase clients.

  a. Set-up a keytab file for the principal and copy the resulting keytab files to where the client daemon will execute.

     Ensure that you make this file readable only to the user account under which the daemon will run.

2. On every HBase client, add the following properties to the `hbase-site.xml` file:

```
<property>
   <name>hbase.security.authentication</name>
    <value>kerberos</value>
</property>
```

Note

The client environment must be logged in to Kerberos from KDC or keytab via the `kinit` command before communication with the HBase cluster is possible. Note that the client will not be able to communicate with the cluster if the `hbase.security.authentication` property in the client- and server-side site files fails to match.

```
<property>
        <name>hbase.rpc.engine</name>
        <value>org.apache.hadoop.hbase.ipc.SecureRpcEngine</value>

</property>
```

13.2.3.2.5. Optional: Configure Client-Side Operation For Secure Operation - Thrift Gateway

Add the following to the `hbase-site.xml` file for every Thrift gateway:

```
  <property>
    <name>hbase.thrift.keytab.file</name>
    <value>/etc/hbase/conf/hbase.keytab</value>
  </property>
  <property>
    <name>hbase.thrift.kerberos.principal</name>
    <value>$USER/_HOST@HADOOP.LOCALDOMAIN</value>
  </property>
```

Substitute the appropriate credential and keytab for *$USER* and *$KEYTAB* respectively.

The Thrift gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the Thrift gateway itself. All client access via the Thrift gateway will use the Thrift gateway's credential and have its privilege.

13.2.3.2.6. Optional: Configure Client-Side Operation For Secure Operation - REST Gateway

Add the following to the hbase-site.xml file for every REST gateway:

```
<property>
  <name>hbase.rest.keytab.file</name>
  <value>$KEYTAB</value>
</property>
<property>
  <name>hbase.rest.kerberos.principal</name>
  <value>$USER/_HOST@HADOOP.LOCALDOMAIN</value>
</property>
```

Substitute the appropriate credential and keytab for *$USER* and *$KEYTAB* respectively.

The REST gateway will authenticate with HBase using the supplied credential. No authentication will be performed by the REST gateway itself. All client access via the REST gateway will use the REST gateway's credential and have its privilege.

13.2.3.2.7. Configure HBase for Access Control Lists (ACL)

Use the following instructions to configure HBase for ACL:

1. Kinit as HBase user.

   a. Create a keytab for princpal `hbase@REALM` and store it in the `hbase.headless.keytab` file. Refer to the instructions provided here for creating principal and keytab file.

   b. Kinit as HBase user. Execute the following command on your HBase Master:

   ```
   kinit -kt hbase.headless.keytab hbase
   ```

2. Start the HBase shell. On the HBase Master host machine, execute the following command:

   ```
   hbase shell
   ```

3. Set ACLs using HBase shell:

   ```
   grant '$USER', '$permissions'
   ```

   where

   - *$USER* is any user responsible for create/update/delete operations in HBase.

     

     Note

     You must set the ACLs for all those users who will be responsible for create/update/delete operations in HBase.

   - *`$permissions`* is zero or more letters from the set "RWCA": READ('R'), WRITE('W'), CREATE('C'), ADMIN('A').

13.2.3.3. Configure Secure Hive

Hive Metastore supports Kerberos authentication for Thrift clients only. HiveServer does not support Kerberos authentication for any clients.

Edit the `hive-site.xml` file on your Hive Metastore host machine to modify the following properties:

```
<property>
        <name>hive.metastore.sasl.enabled</name>
        <value>true</value>
        <description>If true, the metastore thrift interface will be secured
 with
        SASL.
        Clients must authenticate with Kerberos.</description>
</property>
```

```
<property>
        <name>hive.metastore.kerberos.keytab.file</name>
        <value>/etc/security/keytabs/hive.service.keytab</value>
        <description>The path to the Kerberos Keytab file containing the
        metastore thrift server's service principal.</description>
</property>
```

```
<property>
        <name>hive.metastore.kerberos.principal</name>
        <value>hive/_HOST@EXAMPLE.COM</value>
        <description>The service principal for the metastore thrift server.
 The special string _HOST will be replaced automatically with the correct
 hostname.</description>
</property>
```

```
< property>
   <name>hive.server2.authentication</name>
    <value>KERBEROS</value>
    <description>Authentication type </description>
</property>
```

```
<property>
   <name>hive.server2.authentication.kerberos.principal</name>
    <value>hive/_HOST@EXAMPLE.COM</value>
    <description>The service principal for the HiveServer2. If _HOST is used
 as the hostname portion, it will be replaced with the actual hostname of the
 running instance.</description>
</property>
```

```
<property>
    <name>hive.server2.authentication.kerberos.keytab</name>
    <value>/etc/security/keytabs/hive.service.keytab</value>
    <description>The keytab for the HiveServer2 service principal</
description>
</property>
```

13.2.3.4. Configure Secure Oozie

Edit the `oozie-site.xml` file, to add the following information:

```
<property>
    <name>oozie.service.AuthorizationService.security.enabled</name>
    <value>true</value>
    <description>Specifies whether security (user name/admin role) is enabled
 or not.
          If it is disabled any user can manage the Oozie system and manage
 any job.</description>
</property>

<property>
    <name>oozie.service.HadoopAccessorService.kerberos.enabled</name>
    <value>true</value>
    <description>Indicates if Oozie is configured to use Kerberos</
description>
</property>

<property>
    <name>local.realm </name>
    <value>EXAMPLE.COM </value>
    <description>Kerberos Realm used by Oozie and Hadoop. Using 'local.realm'
 to be
          aligned with Hadoop configuration</description>
</property>

<property>
    <name>oozie.service.HadoopAccessorService.keytab.file </name>
    <value>/etc/security/keytabs/oozie.service.keytab</value>
    <description>The keytab for the Oozie service principal.</description>
</property>

<property>
    <name>oozie.service.HadoopAccessorService.kerberos.principal</name>
    <value>$OOZIE_PRINCIPAL/_HOSTl@EXAMPLE.COM </value>
    <description>Kerberos principal for Oozie service</description>
</property>

<property>
    <name>oozie.authentication.type</name>
    <value>kerberos</value>
    <description>Authentication type</description>
</property>

<property>
    <name>oozie.authentication.kerberos.principal</name>
    <value>$HTTP_USER/_HOST@EXAMPLE.COM</value>
    <description>Whitelisted job tracker for Oozie service</description>
</property>

<property>
    <name> oozie.authentication.kerberos.keytab</name>
    <value>/etc/security/keytabs/spnego.service.keytab</value>
    <description>Location of the Oozie user keytab file.</description>
</property>

<property>
    <name>oozie.service.HadoopAccessorService.nameNode.whitelist</name>
    <value/>
    <description/>
</property>
```

```
<property>
    <name>oozie.authentication.kerberos.name.rules</name>
    <value><value>
        RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/$MAPRED_USER/
        RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/$HDFS_USER/
        RULE:[2:$1@$0](hbase@.*EXAMPLE.COM)s/.*/$HBASE_USER/
        RULE:[2:$1@$0](hbase@.*EXAMPLE.COM)s/.*/$HBASE_USER/
        DEFAULT</value>
    <description>The mapping from Kerberos principal names to local service
 user names.
                    </description>
</property>
```

For mapping from Kerberos principal names to local OS user names, see Creating Mappings Between Principals and UNIX Usernames.

13.2.3.5. Configure Secure WebHCat

Edit `webhcat-site.xml` file, to add the following information:

```
</property>
    <name>templeton.kerberos.principal</name>
    <value>HTTP/_HOST@EXAMPLE.COM</value>
    <description/>
<property>
```

```
<property>
    <name>templeton.kerberos.keytab</name>
    <value>/etc/security/keytabs/spnego.service.keytab</value>
    <description/>
</property>
```

```
<property>
    <name>templeton.kerberos.secret</name>
    <value>secret</value>
    <description/>
</property>
```

```
<property>
    <name>templeton.kerberos.properties</name>
    <value>hive.metastore.local=false,hive.metastore.uris=thrift://
MetastoreHost_FQDN:9083,hive.metastore.sasl.enabled=true,hive.metastore.
execute.setugi= true,hive.exec.mode.local.auto=false,hive.metastore.kerberos.
principal=$HIVE_PRINCIPAL/_HOST@EXAMPLE.COM"</value>
         <description/>
</property>
```

14. Uninstalling HDP

Use the following instructions to uninstall HDP:

1. Stop all the services using the instructions provided here.
2. If HBase and ZooKeeper are installed, execute the following commands on all the cluster nodes:

```
rm -f /usr/share/hbase/lib/zookeeper-$version.jar
rm -rf $ZOOKEEPER_PID_DIR/*.pid
rm -rf $HBASE_PID_DIR/*.pid
```

3. If HCatalog is installed, execute the following command on all the cluster nodes:

```
yum remove hcatalog\*
```

4. If Hive is installed, execute the following command on all the cluster nodes:

```
yum remove hive\*
```

5. If Tez is installed, execute the following command on all the cluster nodes:

```
yum remove tez
```

6. If HBase is installed, execute the following command on all the cluster nodes:

```
yum remove hbase\*
```

7. If ZooKeeper is installed, execute the following command on all the cluster nodes:

```
yum remove zookeeper\*
```

8. If Oozie is installed, execute the following command on all the cluster nodes:

```
yum remove oozie\*
```

9. If Pig is installed, execute the following command on all the cluster nodes:

```
yum remove pig\*
```

10. If compression libraries are installed, execute the following command on all the cluster nodes:

```
yum remove snappy\*
yum remove hadoop-lzo\*
```

11. Uninstall Hadoop. Execute the following command on all the cluster nodes:

```
yum remove hadoop\*
```

12. Uninstall ExtJS libraries and MySQL connector. Execute the following command on all the cluster nodes:

```
yum remove extjs-2.2-1 mysql-connector-java-5.0.8-1\*
```

13. Delete Hadoop directories.

```
rm -rf $HADOOP_HOME
```

15. Appendix: Tarballs

The following provides individual links to the Apache structured tarball files for the projects included with Hortonworks Data Platform are listed in the following sections:

- RHEL 5 and CentOS 5
- RHEL 6 and CentOS 6
- SUSE Enterprise Linux 11

15.1. RHEL 5 and CentOS 5

Table 15.1. RHEL/CentOS 5

Project	Download
Hadoop	hadoop-1.2.0.1.3.0.0-107.tar.gz
Pig	pig-0.11.1.1.3.0.0-107.tar.gz
Hive and HCatalog	hive-0.11.0.1.3.0.0-107.tar.gz hcatalog-0.11.0.1.3.0.0-107.tar.gz
Oozie	oozie-3.3.2.1.3.0.0-107-distro.tar.gz
HBase and ZooKeeper	hbase-0.94.6.1.3.0.0-107-security.tar.gz zookeeper-3.4.5.1.3.0.0-107.tar.gz
Sqoop	sqoop-1.4.3.1.3.0.0-107.bin__hadoop-1.2.0.1.3.0.0-107.tar.gz
Flume	apache-flume-1.3.1.1.3.0.0-107-bin.tar.gz
Mahout	mahout-distribution-0.7.0.1.3.0.0-107.tar.gz

15.2. RHEL 6 and CentOS 6

Table 15.2. RHEL/CentOS 6

Project	Download
Hadoop	hadoop-1.2.0.1.3.0.0-107.tar.gz
Pig	pig-0.11.1.1.3.0.0-107.tar.gz
Hive and HCatalog	hive-0.11.0.1.3.0.0-107.tar.gz hcatalog-0.11.0.1.3.0.0-107.tar.gz
Oozie	oozie-3.3.2.1.3.0.0-107-distro.tar.gz
HBase and ZooKeeper	hbase-0.94.6.1.3.0.0-107-security.tar.gz zookeeper-3.4.5.1.3.0.0-107.tar.gz
Sqoop	sqoop-1.4.3.1.3.0.0-107.bin__hadoop-1.2.0.1.3.0.0-107.tar.gz
Flume	apache-flume-1.3.1.1.3.0.0-107-bin.tar.gz
Mahout	mahout-distribution-0.7.0.1.3.0.0-107.tar.gz

15.3. SUSE Enterprise Linux 11

Table 15.3. SLES 11

Project	Download
Hadoop	hadoop-1.2.0.1.3.0.0-107.tar.gz
Pig	pig-0.11.1.1.3.0.0-107.tar.gz
Hive and HCatalog	hive-0.11.0.1.3.0.0-107.tar.gz hcatalog-0.11.0.1.3.0.0-107.tar.gz
Oozie	oozie-3.3.2.1.3.0.0-107-distro.tar.gz
HBase and ZooKeeper	hbase-0.94.6.1.3.0.0-107-security.tar.gz zookeeper-3.4.5.1.3.0.0-107.tar.gz
Sqoop	sqoop-1.4.3.1.3.0.0-107.bin__hadoop-1.2.0.1.3.0.0-107.tar.gz
Flume	apache-flume-1.3.1.1.3.0.0-107-bin.tar.gz
Mahout	mahout-distribution-0.7.0.1.3.0.0-107.tar.gz