Hortonworks Data Platform

Installing HDP Manually

(Jan 22, 2015)

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.

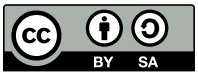


Table of Contents

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. Use the following instructions before you deploy Hadoop cluster using HDP:

1. Understand the basics
2. Meet minimum system requirements
3. Configure the remote repositories
4. Decide on deployment type
5. Collect information
6. Prepare the environment
7. Optional - Install MySQL
8. Download companion files
9. Define environment parameters
10. Optional - Create system users and groups
11. Determine YARN and MapReduce Memory Configuration Settings
12. Allocate Adequate Log Space for HDP

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.
  - **Apache Hadoop YARN**: YARN is a general-purpose, distributed, application management framework that supersedes the classic Apache Hadoop MapReduce framework for processing data in Hadoop clusters. The fundamental idea of YARN is to split up the two major responsibilities of the JobTracker i.e. resource management and job scheduling/monitoring, into separate daemons: a global **ResourceManager** and per-application **ApplicationMaster** (AM). The ResourceManager and per-node slave, the **NodeManager** (NM), form the new, and generic, system for managing applications in a distributed manner. The ResourceManager is the ultimate authority that arbitrates resources among all the applications in the system. The per-application ApplicationMaster is, in effect, a framework specific entity and is tasked with negotiating resources from the ResourceManager and working with the NodeManager(s) to execute and monitor the component tasks.

  - **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-like queries using HiveQL, the tool's native language, that can be compiled into sequences of MapReduce programs.

  - **Tez**: A general-purpose, highly customizable framework that creates simplifies data-processing tasks across both small scale (low-latency) and large-scale (high throughput) workloads in Hadoop.

  - **Apache HCatalog**: A metadata abstraction layer that insulates users and scripts from how and where data is physically stored.

  - **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 Apache components designed to ease working with Core Hadoop.

  - **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.

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
- Database Requirements
- JDK Recommendations

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: Suggested Hardware for a Typical Hadoop Cluster.

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 Oracle Linux 5 or 6
- 64-bit SUSE Linux Enterprise Server (SLES) 11, SP1
- 64-bit Ubuntu Precise (12.04)

1.2.3. Software Requirements

On each of your hosts:

- yum [for RHEL or CentOS]
- zypper [for SLES]
- php_curl [for SLES]
- apt-get [for Ubuntu]
- rpm
- scp
- curl
- wget
- unzip
- tar

1.2.4. Database Requirements

To use external database for Hive or Oozie metastore, have a MySQL, Oracle, or PostgreSQL database deployed and available.

By default, Hive and Oozie use Derby database for its metastore. To use an external database for Hive and Oozie metastore, ensure that a MySQL database is deployed and available.

- You can choose to use a current instance of MySQL or install a new instance for its use. For more information, see Install MySQL (Optional).

- For instructions on configuring an existing Oracle database instance, see here [4].

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 [4].
- Ensure that your database administrator creates the following databases and users.
  - For Hive, ensure that your database administrator creates `hive_dbname`, `hive_dbuser`, and `hive_dbpasswd`.
  - For Oozie, ensure that your database administrator creates `oozie_dbname`, `oozie_dbuser`, and `oozie_dbpasswd`.

Note

For instructions on creating users for MySQL, see here.

Instructions to configure an Oracle database

- Run following SQL script against your Hive schema:

```
/usr/lib/hive/scripts/metastore/upgrade/oracle/hive-schema-0.12.0.oracle.sql
```

Instructions to deploy and configure a 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
     ```

   - For Ubuntu:

     ```
     apt-get 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. Run the following SQL script against your Hive schema:

```
/usr/lib/hive/scripts/metastore/upgrade/postgres/hive-schema-0.12.0.
postgres.sql
```

1.2.5. JDK Requirements

Your system must have the correct JDK installed on all the nodes of the cluster. HDP supports the following JDKs.

- Oracle JDK 1.6 update 31 64-bit
- Oracle JDK 7 64-bit
- Open JDK 7 64-bit

1.2.5.1. 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. From your browser window, go to `http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u31-oth-JPR`.

   Accept the license agreement and download `jdk-6u31-linux-x64.bin` to a temporary directory (**$JDK_download_directory**).

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_H0ME 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
```

8. Verify if Java is installed in your environment. Execute the following from the command line console:

```
java -version
```

   You should see the following output:

```
java version "1.6.0_31"
Java(TM) SE Runtime Environment (build 1.6.0_31-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
```

1.2.5.2. Oracle JDK 7 update 40

Use the following instructions to manually install JDK 7:

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

```
java -version
```

2. (Optional) Uninstall the Java package if the JDK version is less than 7.

```
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-7u40-linux-x64.tar.gz) from the Oracle download site. From your browser window, go to `http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html`.

   Accept the license agreement and download `jdk-7u40-linux-x64.tar.gz` to a temporary directory (**$JDK_download_directory**).

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

```
mkdir /usr/jdk1.7.0_40
cd /usr/jdk1.7.0_40
chmod u+x $JDK_download_directory/jdk-7u40-linux-x64.bin
./$JDK_download_directory/jdk-7u40-linux-x64.bin
```

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

```
mkdir /usr/java
ln -s /usr/jdk1.7.0_40/jdk1.7.0_40 /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
```

8. Verify if Java is installed in your environment. Execute the following from the command line console:

```
java -version
```

   You should see the following output:

```
java version "1.7.0_40"
Java(TM) SE Runtime Environment (build 1.7.0_40-b04)
Java HotSpot(TM) 64-Bit Server VM (build 20.6-b01, mixed mode)
```

1.2.5.3. Open JDK 7

Use the following instructions to manually install Open JDK 7:

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

```
java -version
```

2. (Optional) Uninstall the Java package if the JDK version is less than 7.

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

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

```
which java
```

4. Download Open JDK 7 RPMs. From the command-line, run:

```
yum install java-1.7.0-openjdk java-1.7.0-openjdk-devel
```

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

```
mkdir /usr/java
ln -s /usr/openjdk1.7.0/openjdk1.7.0 /usr/java/default
ln -s /usr/java/default/bin/java /usr/bin/java
```

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

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

7. Verify if Java is installed in your environment. Execute the following from the command-line console:

```
java -version
```

You should see output similar to the following:

```
openjdk version "1.7.0"
OpenJDK Runtime Environment (build 1.7.0)
OpenJDK Client VM (build 20.6-b01, mixed mode)
```

1.2.6. 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. Configure the Remote Repositories

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. Download the yum repo configuration files `hdp.repo` and `ambari.repo`. On your local mirror server, execute the following command:

   - For RHEL/CentOS 5:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.0.
13.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
wget -nv http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/
1.4.4.23/ambari.repo -O /etc/yum.repos.d/ambari.repo
```

- For RHEL/CentOS 6:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.0.
13.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
wget -nv http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/
1.4.4.23/ambari.repo -O /etc/yum.repos.d/ambari.repo
```

- For SLES:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/suse11/2.x/updates/2.0.
13.0/hdp.repo -O /etc/zypp.repos.d/hdp.repo
wget -nv http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.
4.4.23/ambari.repo -O /etc/zypp.repos.d/ambari.repo
```

- For Ubuntu:

```
wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/hdp.list -O /
etc/apt/sources.list.d/hdp.list
```

2. For Ubuntu hosts, add the gpg keys as the root user:

```
gpg --keyserver pgp.mit.edu --recv-keys B9733A7A07513CAD
gpg -a --export 07513CAD | apt-key add -
```

3. Confirm the HDP repository is configured.

- For RHEL/CentOS/Oracle Linux:

```
yum repolist
```

You should see something like this. Verify that you have HDP-2.0.13.0 directory:

```

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
HDP-2.0.13.0 Hortonworks Data Platform Version - HDP-2.0.13.0 enabled: 53
```

- For SLES:

```
zypper repos
```

- For Ubuntu:

```
apt-get update
```

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. 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 instance, if you install Hive/HCatalog.

Note

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

1.6. Prepare the Environment

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

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

1.6.1. 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 xserver. Use the following instructions to enable NTP for your cluster:

1. Configure NTP clients. Execute the following command on all the nodes in your cluster:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install ntp
     ```

   - For SLES:

     ```
     zypper install ntp
     ```

   - For Ubuntu:

     ```
     apt-get install ntp
     ```

2. Enable the service. Execute the following command on all the nodes in your cluster:

   ```
   chkconfig ntpd on
   ```

3. Start the NTP. Execute the following command on all the nodes in your cluster:

   ```
   /etc/init.d/ntpd start
   ```

4. You can use the existing NTP server in your environment. Configure the firewall on the local NTP server to enable UDP input traffic on port 123 and replace 192.168.1.0/24 with the ip addresses in the cluster. See the following sample rule:

```
# iptables -A RH-Firewall-1-INPUT -s 192.168.1.0/24 -m state --state NEW -p
 udp --dport 123 -j ACCEPT
```

Restart iptables. Execute the following command on all the nodes in your cluster:

```
# iptables service iptables restart
```

Configure clients to use the local NTP server. Edit the `/etc/ntp.conf` and add the following line:

```
server $LOCAL_SERVER_IP OR HOSTNAME
```

1.6.2. 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.

Use the following instructions to check DNS for all the host machines in your cluster:

1. Forward lookup checking.

   For example, for domain `localdomain` that contains host with name `host01` and IP address `192.168.0.10`, execute the following command:

```
nslookup host01
```

   You should see a message similar to the following:

```
Name:        host01.localdomain
Address: 192.168.0.10
```

2. Reverse lookup checking.

   For example, for domain `localdomain` that contains host with name `host01` and IP address `192.168.0.10`, execute the following command:

```
nslookup 192.168.0.10
```

   You should see a message similar to the following:

```
 10.0.168.192.in-addr.arpa        name = host01.localdomain.
```

If you do not receive valid responses (as shown above), you should set up DNS zone in your cluster or configure host files on each host of the cluster using one of the following options:

- **Option I:** Configure hosts file on each node of the cluster.

  For all nodes of cluster, add to the `/etc/hosts` file key-value pairs like the following:

```
192.168.0.11        host01
```

- **Option II:** Configuring DNS using BIND nameserver.

  The following instructions, use the example values given below:

```
Example values:
domain name: “localdomain”
nameserver: “host01”/192.168.0.11
hosts: “host02”/192.168.0.12, “host02”/192.168.0.12
```

  1. Install BIND packages:

```
yum install bind
yum install bind-libs
yum install bind-utils
```

  2. Initiate service

```
chkconfig named on
```

  3. Configure files. Add the following lines for the example values given above (ensure that you modify these for your environment) :

     - Edit the `/etc/resolv.conf` (for all nodes in cluster) and add the following lines:

```
domain localdomain
search localdomain
nameserver 192.168.0.11
```

     - Edit the `/etc/named.conf` (for all nodes in cluster) and add the following lines:

```
listen-on port 53 { any; };//by default it is opened only for localhost
 ...
zone "localdomain" {
 type master;
 notify no;
 allow-query { any; };
 file "named-forw.zone";
 };
 zone "0.168.192.in-addr.arpa" {
  type master;
  notify no;
  allow-query { any; };
  file "named-rev.zone";
};
```

     - Edit the `named-forw.zone` as shown in the following sample forward zone configuration file:

```
$TTL 3D
@        SOA   host01.localdomain.root.localdomain
(201306030;3600;3600;3600;3600)
NS        host01            ; Nameserver Address
localhost        IN        A        127.0.0.1
host01               IN        A        192.168.0.11
host02               IN        A        192.168.0.12
host03               IN        A        192.168.0.13
```

- Edit the `named-rev.zone` as shown in the following sample reverse zone configuration file:

```
$TTL 3D
@        SOA        host01.localdomain.root.localdomain.
(201306031;28800;2H;4W;1D);
NS        host01.localdomain.; Nameserver Address
11        IN        PTR        host01.localdomain.
12        IN        PTR        host02.localdomain.
13        IN        PTR        host03.localdomain.
```

4. Restart bind service.

```
/etc/init.d/named restart
```

5. Add rules to firewall.

```
iptables -A INPUT -p udp -m state --state NEW --dport 53 -j ACCEPT
iptables -A INPUT -p tcp -m state --state NEW --dport 53 -j ACCEPT
service iptables save
service iptables restart
```

Alternatively, you can also allow traffic over DNS port (53) using `system-config-firewall` utility.

1.6.3. Disable SELinux

Security-Enhanced (SE) Linux feature should be disabled during installation process.

1. Check state of SELinux. On all the host machines, execute the following command:

```
getenforce
```

If the result is `permissive` or `disabled`, no further actions are required, else proceed to step 2.

2. Disable SELinux either temporarily for each session or permanently.

- **Option I:** Disable SELinux temporarily by executing the following command:

```
setenforce 0
```

- **Option II:** Disable SELinux permanently in the `/etc/sysconfig/selinux` file by changing the value of `SELINUX` field to `permissive` or `disabled`. Restart your system.

1.6.4. Disable IPTables

On all the RHEL/CentOS host machines, execute the following command to disable IPTables:

```
chkconfig iptables off
/etc/init.d/iptables stop
```

On Ubuntu host machines, execute the following command to disable IPTables:

```
service ufw stop
```

1.7. [Optional] Install MySQL

If you are installing Hive and HCatalog services, you need a MySQL database instance to store metadata information. You can either use an existing MySQL instance or install a new instance of MySQL manually. To install a new instance:

1. Connect to the host machine you plan to use for Hive and HCatalog.
2. Install MySQL server. From a terminal window, type:

   For RHEL/CentOS:

   ```
   yum install mysql-server
   ```

3. Start the instance.

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

4. Set the `root` user password.

   ```
   mysqladmin -u root -p'{password}' password $mysqlpassword
   ```

5. Remove unnecessary information from log and STDOUT.

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

6. As `root`, use mysql (or other client tool) to create the "dbuser" and grant it adequate privileges. This user provides access to the Hive metastore.

   ```
   CREATE USER '$dbusername'@'localhost' IDENTIFIED BY '$dbuserpassword';
   GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'localhost';
   CREATE USER 'dbuser'@'%' IDENTIFIED BY 'dbuserpassword';
   GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'%';
   FLUSH PRIVILEGES;
   ```

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

   ```
   mysql -u dbuser -p $dbuserpassword
   ```

8. Install the MySQL connector JAR file.

   - For RHEL/CentOS/Oracle Linux:

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

   - For SLES:

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

   - For Ubuntu:

     ```
     apt-get install mysql-connector-java*
     ```

1.8. Download Companion Files

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

```
wget http://public-repo-1.hortonworks.com/HDP/tools/2.0.6.0/
hdp_manual_install_rpm_helper_files-2.0.13.001.tar.gz
```

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

The following provides a snapshot of a sample script file to create Hadoop directories. This sample script file sources the files included in Companion Files.

```
#!/bin/bash
./users.sh
./directories.sh

echo "Create datanode local dir"
mkdir -p $DFS_DATA_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIR;
chmod -R 750 $DFS_DATA_DIR;

echo "Create yarn local dir"
mkdir -p $YARN_LOCAL_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_DIR;
chmod -R 755 $YARN_LOCAL_DIR;

echo "Create yarn local log dir"
mkdir -p $YARN_LOCAL_LOG_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_LOG_DIR;
chmod -R 755 $YARN_LOCAL_LOG_DIR;
```

1.9. Define Environment Parameters

You need to set up specific users and directories for your HDP installation using the following instructions:

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 companion files that 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 1.1. Define Users and Groups for Systems

Parameter	Definition
HDFS_USER	User owning the HDFS services. For example, hdfs.
YARN_USER	User owning the YARN services. For example, yarn.
ZOOKEEPER_USER	User owning the ZooKeeper services. For example, zookeeper.
HIVE_USER	User owning the Hive services. For example, hive.
WEBHCAT_USER	User owning the WebHCat services. For example, hcat.
HBASE_USER	User owning the HBase services. For example, hbase.
PIG_USER	User owning the Pig services. For example, pig.
HADOOP_GROUP	A common group shared by services. For example, hadoop.

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 companion files that 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 1.2. 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.

Hadoop Service	Parameter	Definition

		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 *$HDFS_USER*. For example, `/var/log/hadoop/hdfs` where `hdfs` is the *$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`

YARN	`YARN_LOCAL_DIR`	Space separated list of directories where YARN should store temporary data.
		For example,
		`/grid/hadoop/yarn`
		`/grid1/hadoop/yarn`
		`/grid2/hadoop/yarn`.

YARN	`YARN_LOG_DIR`	Directory for storing the YARN logs. For example, `/var/log/hadoop/yarn`. This directory name is a combination of a directory and the *$YARN_USER*. In the example `yarn` is the *$YARN_USER*.
YARN	`YARN_PID_DIR`	Directory for storing the YARN process ID. For example, `/var/run/hadoop/yarn`. This directory name is a combination of a directory and the *$YARN_USER*. In the example, `yarn` is the *$YARN_USER*.

Hadoop Service	Parameter	Definition
MapReduce	MAPRED_LOG_DIR	Directory for storing the JobHistory Server 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*

Table 1.3. 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.
Pig	PIG_LOG_DIR	Directory to store the Pig logs. For example, /var/log/pig.
Pig	PIG_PID_DIR	Directory to store the Pig process ID. For example, /var/run/pig.
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.
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, var/log/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.

Hadoop Service	Parameter	Definition
ZooKeeper	ZOOKEEPER_DATA_DIR	Directory where ZooKeeper will store data. For example, /grid/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.

1.10. [Optional] Create System Users and Groups

In general Hadoop services should be owned by specific users and not by root or application users. The table below shows the typical users for Hadoop services. If you choose to install the HDP components using the RPMs, these users will automatically be set up.

If you do not install with the RPMs, or want different users, then you must identify the users that you want for your Hadoop services and the common Hadoop group and create these accounts on your system.

Table 1.4. Typical System Users and Groups

Hadoop Service	User	Group
HDFS	hdfs	hadoop
YARN	yarn	hadoop
MapReduce	mapred	hadoop
Hive	hive	hadoop
Pig	pig	hadoop
HCatalog/WebHCatalog	hcat	hadoop
HBase	hbase	hadoop
ZooKeeper	zookeeper	hadoop
Oozie	oozie	hadoop

1.11. Determine YARN and MapReduce Memory Configuration Settings

1.11.1. Manually Calculate YARN and MapReduce Memory Configuration Settings

This section describes how to manually calculate YARN and MapReduce memory allocation settings based on the node hardware specifications.

YARN takes into account all of the available resources on each machine in the cluster. Based on the available resources, YARN negotiates resource requests from applications (such as MapReduce) running in the cluster. YARN then provides processing capacity to each application by allocating *Containers*. A Container is the basic unit of processing capacity in YARN, and is an encapsulation of resource elements (memory, CPU, etc.).

In a Hadoop cluster, it is vital to balance the usage of memory (RAM), processors (CPU cores) and disks so that processing is not constrained by any one of these cluster resources. As a general recommendation, allowing for two Containers per disk and per core gives the best balance for cluster utilization.

When determining the appropriate YARN and MapReduce memory configurations for a cluster node, start with the available hardware resources. Specifically, note the following values on each node:

- RAM (Amount of memory)
- CORES (Number of CPU cores)
- DISKS (Number of disks)

The total available RAM for YARN and MapReduce should take into account the Reserved Memory. Reserved Memory is the RAM needed by system processes and other Hadoop processes, such as HBase.

Reserved Memory = Reserved for stack memory + Reserved for HBase memory (If HBase is on the same node)

Use the following table to determine the Reserved Memory per node.

Reserved Memory Recommendations

Total Memory per Node	Recommended Reserved System Memory	Recommended Reserved HBase Memory
4 GB	1 GB	1 GB
8 GB	2 GB	1 GB
16 GB	2 GB	2 GB
24 GB	4 GB	4 GB
48 GB	6 GB	8 GB
64 GB	8 GB	8 GB
72 GB	8 GB	8 GB
96 GB	12 GB	16 GB
128 GB	24 GB	24 GB
256 GB	32 GB	32 GB
512 GB	64 GB	64 GB

The next calculation is to determine the maximum number of Containers allowed per node. The following formula can be used:

# of Containers = minimum of (2*CORES, 1.8*DISKS, (Total available RAM) / MIN_CONTAINER_SIZE)

Where MIN_CONTAINER_SIZE is the minimum Container size (in RAM). This value is dependent on the amount of RAM available – in smaller memory nodes, the minimum Container size should also be smaller. The following table outlines the recommended values:

Total RAM per Node	Recommended Minimum Container Size
Less than 4 GB	256 MB
Between 4 GB and 8 GB	512 MB
Between 8 GB and 24 GB	1024 MB
Above 24 GB	2048 MB

The final calculation is to determine the amount of RAM per container:

RAM-per-Container = maximum of (MIN_CONTAINER_SIZE, (Total Available RAM) / Containers))

With these calculations, the YARN and MapReduce configurations can be set:

Configuration File	Configuration Setting	Va
yarn-site.xml	yarn.nodemanager.resource.memory-mb	= C
yarn-site.xml	yarn.scheduler.minimum-allocation-mb	= R
yarn-site.xml	yarn.scheduler.maximum-allocation-mb	= c
mapred-site.xml	mapreduce.map.memory.mb	= R
mapred-site.xml	mapreduce.reduce.memory.mb	= 2
mapred-site.xml	mapreduce.map.java.opts	= 0
mapred-site.xml	mapreduce.reduce.java.opts	= 0
yarn-site.xml (check)	yarn.app.mapreduce.am.resource.mb	= 2
yarn-site.xml (check)	yarn.app.mapreduce.am.command-opts	= 0

Note: After installation, both `yarn-site.xml` and `mapred-site.xml` are located in the `/etc/hadoop/conf` folder.

Examples

Cluster nodes have 12 CPU cores, 48 GB RAM, and 12 disks.

Reserved Memory = 6 GB reserved for system memory + (if HBase) 8 GB for HBase

Min Container size = 2 GB

If there is no HBase:

# of Containers = minimum of (2*12, 1.8* 12, (48-6)/2) = minimum of (24, 21.6, 21) = 21

RAM-per-Container = maximum of (2, (48-6)/21) = maximum of (2, 2) = 2

Configuration	Value Calculation
yarn.nodemanager.resource.memory-mb	= 21 * 2 = 42*1024 MB

yarn.scheduler.minimum-allocation-mb	= 2*1024 MB
yarn.scheduler.maximum-allocation-mb	= 21 * 2 = 42*1024 MB
mapreduce.map.memory.mb	= 2*1024 MB
mapreduce.reduce.memory.mb	= 2 * 2 = 4*1024 MB
mapreduce.map.java.opts	= 0.8 * 2 = 1.6*1024 MB
mapreduce.reduce.java.opts	= 0.8 * 2 * 2 = 3.2*1024 MB
yarn.app.mapreduce.am.resource.mb	= 2 * 2 = 4*1024 MB
yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * 2 = 3.2*1024 MB

If HBase is included:

# of Containers = minimum of (2*12, 1.8* 12, (48-6-8)/2) = minimum of (24, 21.6, 17) = 17

RAM-per-Container = maximum of (2, (48-6-8)/17) = maximum of (2, 2) = 2

Configuration	Value Calculation
yarn.nodemanager.resource.memory-mb	= 17 * 2 = 34*1024 MB
yarn.scheduler.minimum-allocation-mb	= 2*1024 MB
yarn.scheduler.maximum-allocation-mb	= 17 * 2 = 34*1024 MB
mapreduce.map.memory.mb	= 2*1024 MB
mapreduce.reduce.memory.mb	= 2 * 2 = 4*1024 MB
mapreduce.map.java.opts	= 0.8 * 2 = 1.6*1024 MB
mapreduce.reduce.java.opts	= 0.8 * 2 * 2 = 3.2*1024 MB
yarn.app.mapreduce.am.resource.mb	= 2 * 2 = 4*1024 MB
yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * 2 = 3.2*1024 MB

1.11.2. Use the YARN Utility Script to Calculate YARN and MapReduce Memory Configuration Settings

This section describes how to use the `yarn-util.py` Python script to calculate YARN and MapReduce memory allocation settings based on the node hardware specifications. The `yarn-util.py` script is included in the HDP companion files.

Running the Script

To run the `yarn-util.py` script, execute the following command from the folder containing the script:

```
python yarn-util.py <options>
```

With the following options:

Option	Description
-c CORES	The number of cores on each host.
-m MEMORY	The amount of memory on each host in GB.
-d DISKS	The number of disks on each host.
-k HBASE	"True" if HBase is installed, "False" if not.

Note: You can also use the `-h` or `--help` option to display a Help message that describes the options.

Example

Running the following command:

```
python yarn-utils.py -c 16 -m 64 -d 4 -k True
```

Would return:

```
Using cores=16 memory=64GB disks=4 hbase=True
Profile: cores=16 memory=64GB reserved=16GB usableMem=48GB disks=4
Num Container=32
Container Ram=1536MB
Used Ram=48GB
Unused Ram=16GB
yarn.scheduler.minimum-allocation-mb=1536
yarn.scheduler.maximum-allocation-mb=49152
yarn.nodemanager.resource.memory-mb=49152
mapreduce.map.memory.mb=1536
mapreduce.map.java.opts=-Xmx1228m
mapreduce.reduce.memory.mb=3072
mapreduce.reduce.java.opts=-Xmx2457m
yarn.app.mapreduce.am.resource.mb=3072
yarn.app.mapreduce.am.command-opts=-Xmx2457m
mapreduce.task.io.sort.mb=614
```

1.12. Allocate Adequate Log Space for HDP

Logs are an important part of managing and operating your HDP cluster. The directories and disks that you assign for logging in HDP must have enough space to maintain logs during HDP operations. Allocate at least 10GB of free space for any disk you want to use for HDP logging."

2. Installing HDFS and YARN

This section describes how to install the Hadoop Core components, HDFS, YARN, and MapReduce.

Complete the following instructions to install Hadoop Core components:

1. Set Default File and Directory Permissions
2. Install the Hadoop Packages
3. Install Compression Libraries
4. Create Directories

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.

Ensure that the `umask` is set for all terminal sessions that you use during installation.

2.2. Install the Hadoop Packages

Execute the following command on all cluster nodes.

- For RHEL/CentOS/Oracle Linux:

```
yum install hadoop hadoop-hdfs hadoop-libhdfs hadoop-yarn hadoop-mapreduce
 hadoop-client openssl
```

- For SLES:

```
zypper install hadoop hadoop-hdfs hadoop-libhdfs hadoop-yarn hadoop-
mapreduce hadoop-client openssl
```

- For Ubuntu:

```
apt-get install hadoop hadoop-hdfs libhdfs0 libhdfs0-dev hadoop-yarn hadoop-
mapreduce hadoop-client openssl
```

2.3. Install Compression Libraries

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

2.3.1. Install Snappy

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

1. Install Snappy. From a terminal window, type:

   - For RHEL/CentOS/Oracle Linux:

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

   - For SLES:

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

   - For Ubuntu:

     ```
     apt-get install libsnappy1 libsnappy-dev
     ```

2. Make the Snappy libraries available to Hadoop:

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

2.3.2. Install LZO

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

- For RHEL/CentOS/Oracle Linux:

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

- For SLES:

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

- For Ubuntu:

  ```
  apt-get install liblzo2-2 liblzo2-dev hadoop-lzo
  ```

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 bash script files included with the companion files (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 YARN NodeManager 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 run 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:

- *$FS_CHECKPOINT_DIR* is the space separated list of directories where SecondaryNameNode should store the checkpoint image. For example, `/grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/snn /grid2/hadoop/hdfs/snn`.
- *$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 DataNode and YARN NodeManager Local Directories

On all DataNodes, execute the following commands:

```
mkdir -p $DFS_DATA_DIR;
chown -R $HDFS_USER:$HADOOP_GROUP $DFS_DATA_DIR;
chmod -R 750 $DFS_DATA_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 /grid2/hadoop/hdfs/dn`.

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

On the ResourceManager and all DataNodes, execute the following commands:

```
mkdir -p $YARN_LOCAL_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_DIR;
chmod -R 755 $YARN_LOCAL_DIR;
```

where:

- *$YARN_LOCAL_DIR* is the space separated list of directories where YARN should store temporary data. For example, `/grid/hadoop/yarn/local /grid1/hadoop/yarn/local /grid2/hadoop/yarn/local`.
- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

On the ResourceManager and all DataNodes, execute the following commands:

```
mkdir -p $YARN_LOCAL_LOG_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOCAL_LOG_DIR;
chmod -R 755 $YARN_LOCAL_LOG_DIR;
```

where:

- *$YARN_LOCAL_LOG_DIR* is the space separated list of directories where YARN should store temporary data. For example, `/grid/hadoop/yarn/logs /grid1/hadoop/yarn/logs /grid2/hadoop/yarn/local`.
- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.
- *$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;
```

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

```
mkdir -p $YARN_LOG_DIR;
```

```
chown -R $YARN_USER:$HADOOP_GROUP $YARN_LOG_DIR;
chmod -R 755 $YARN_LOG_DIR;
```

where:

- *$YARN_LOG_DIR* is the directory for storing the YARN logs.

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

  For example, `/var/log/hadoop/yarn` where `yarn` is the *$YARN_USER*.

- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.

- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

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

where:

- *$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*.

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

- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

```
mkdir -p $YARN_PID_DIR;
chown -R $YARN_USER:$HADOOP_GROUP $YARN_PID_DIR;
chmod -R 755 $YARN_PID_DIR;
```

where:

- *$YARN_PID_DIR* is the directory for storing the YARN process ID.

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

  For example, `/var/run/hadoop/yarn` where `yarn` is the *$YARN_USER*.

- *$YARN_USER* is the user owning the YARN services. For example, `yarn`.

- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

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

where:

- *$MAPRED_LOG_DIR* is the directory for storing the JobHistory Server logs.

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

  For example, `/var/log/hadoop/mapred` where `mapred` is the *$MAPRED_USER*.

- *$MAPRED_USER* is the user owning the MAPRED services. For example, `mapred`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

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

where:

- *$MAPRED_PID_DIR* is the directory for storing the JobHistory Server pid.

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

  For example, `/var/run/hadoop/mapred` where `mapred` is the *$MAPRED_USER*.
- *$MAPRED_USER* is the user owning the MAPRED services. For example, `mapred`.
- *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

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 bash script files included with the companion files (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. Extract the core Hadoop configuration files to a temporary directory.

   The files are located in the `configuration_files/core_hadoop`directory where you decompressed the companion files.

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` and modify the following properties:

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

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

```
<property>
 <name>dfs.namenode.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.datanode.data.dir</name>
 <value>file:///grid/hadoop/hdfs/dn, file:///grid1/hadoop/hdfs/dn</value>

 <description>Comma separated list of paths. Use the list of directories
 from $DFS_DATA_DIR.
                For example, file:///grid/hadoop/hdfs/dn, file:///grid1/
hadoop/hdfs/dn.</description>
                </property>
```

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

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

```
<property>
   <name>dfs.namenode.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>
```

Note

The value of NameNode new generation size should be 1/8 of maximum heap size (`-Xmx`). Ensure that you check the default setting for your environment.

To change the default value:

i. Edit the `/etc/hadoop/conf/hadoop-env.sh` file.

ii. Change the value of the `-XX:MaxnewSize` parameter to 1/8th the value of the maximum heap size (`-Xmx`) parameter.

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

```
<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>$resourcemanager.full.hostname:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>$resourcemanager.full.hostname:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.address</name>
 <value>$resourcemanager.full.hostname:8050</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>$resourcemanager.full.hostname:8141</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.nodemanager.local-dirs</name>
 <value>/grid/hadoop/hdfs/yarn/local,/grid1/hadoop/hdfs/yarn/local</
value>
 <description>Comma separated list of paths. Use the list of directories
 from $YARN_LOCAL_DIR.
               For example, /grid/hadoop/hdfs/yarn/local,/grid1/hadoop/
hdfs/yarn/local.</description>
</property>
```

```
<property>
 <name>yarn.nodemanager.log-dirs</name>
 <value>/gird/hadoop/hdfs/yarn/logs</value>
 <description>Use the list of directories from $YARN_LOCAL_LOG_DIR.
               For example, /grid/hadoop/yarn/logs /grid1/hadoop/yarn/
logs /grid2/hadoop/yarn/logs</description>
</property>
```

```
<property>
 <name>yarn.log.server.url</name>
 <value>http://$jobhistoryserver.full.hostname:19888/jobhistory/logs/</
value>
 <description>URL for job history server</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>$resourcemanager.full.hostname:8088</value>
 <description>URL for job history server</description>
</property>
```

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

```
<property>
 <name>mapreduce.jobhistory.address</name>
 <value>$jobhistoryserver.full.hostname:10020</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>
```

```
<property>
 <name>mapreduce.jobhistory.webapp.address</name>
 <value>$jobhistoryserver.full.hostname:19888</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>
```

4. Optional: Configure MapReduce to use Snappy Compression

   In order to enable Snappy compression for MapReduce jobs, edit core-site.xml and mapred-site.xml.

   a. Add the following properties to `mapred-site.xml`:

```
<property>
<name>mapreduce.admin.map.child.java.opts</name>
```

```
<value>-server -XX:NewRatio=8 -Djava.library.path=/usr/lib/hadoop/lib/
native/ -Djava.net.preferIPv4Stack=true</value>
<final>true</final>
</property>
<property>
<name>mapreduce.admin.reduce.child.java.opts</name>
<value>-server -XX:NewRatio=8 -Djava.library.path=/usr/lib/hadoop/lib/
native/ -Djava.net.preferIPv4Stack=true</value>
<final>true</final>
</property>
```

b. Add the SnappyCodec to the codecs list in core-site.xml:

```
<property>
<name>io.compression.codecs</name>
<value>org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.
compress.DefaultCodec,org.apache.hadoop.io.compress.SnappyCodec</value>
</property>
```

5. Optional: Replace the default memory configuration settings in `yarn-site.xml` and `mapred-site.xml` with the YARN and MapReduce memory configuration settings you calculated previously.

6. Copy the configuration files.

   a. On all hosts in your cluster, create the Hadoop configuration directory:

   ```
   rm -r $HADOOP_CONF_DIR
   mkdir -p $HADOOP_CONF_DIR
   ```

   where *$HADOOP_CONF_DIR* is the directory for storing the Hadoop configuration files.

   For example, `/etc/hadoop/conf`.

   b. Copy all the configuration files to *$HADOOP_CONF_DIR*.

   c. Set appropriate permissions:

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

   where:

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

4. Validating the Core Hadoop Installation

Use the following instructions to start core Hadoop and perform the smoke tests:

1. Format and Start HDFS
2. Smoke Test HDFS
3. Start YARN
4. Start MapReduce JobHistory Server
5. Smoke Test MapReduce

4.1. Format and Start HDFS

1. Execute these commands on the NameNode host machine:

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

2. Execute these commands on the SecondaryNameNode:

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

3. Execute these commands on all DataNodes:

```
su $HDFS_USER
/usr/lib/hadoop/sbin/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`.

4.2. Smoke Test HDFS

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

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

2. Create hdfs user directory in HDFS:

```
su $HDFS_USER
hadoop fs -mkdir -p /user/hdfs
```

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

```
su $HDFS_USER
hadoop fs -copyFromLocal /etc/passwd passwd
hadoop fs -ls
```

4. Test browsing HDFS:

```
http://$datanode.full.hostname:50075/browseDirectory.jsp?namenodeInfoPort=
50070&dir=/&nnaddr=$namenode.full.hostname:8020
```

4.3. Start YARN

1. Execute these commands from the ResourceManager server:

```
<login as $YARN_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start
 resourcemanager
```

2. Execute these commands from all NodeManager nodes:

```
<login as $YARN_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config $HADOOP_CONF_DIR start
 nodemanager
```

where:

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

4.4. Start MapReduce JobHistory Server

1. Change permissions on the container-executor file.

```
chown -R root:hadoop /usr/lib/hadoop-yarn/bin/container-executor
chmod -R 6050 /usr/lib/hadoop-yarn/bin/container-executor
```

Note

If these permissions are not set, the healthcheck script will return an errror stating that the datanode is UNHEALTHY.

2. Execute these commands from the JobHistory server to set up directories on HDFS :

```
su $HDFS_USER
hadoop fs -mkdir -p /mr-history/tmp
hadoop fs -chmod -R 1777 /mr-history/tmp
hadoop fs -mkdir -p /mr-history/done
hadoop fs -chmod -R 1777 /mr-history/done
hadoop fs -chown -R $MAPRED_USER:$HDFS_USER /mr-history

hadoop fs -mkdir -p /app-logs
```

```
hadoop fs -chmod -R 1777 /app-logs
hadoop fs -chown yarn /app-logs
```

3. Execute these commands from the JobHistory server:

```
<login as $MAPRED_USER>
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec/
/usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh --
config $HADOOP_CONF_DIR start historyserver
```

where:

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

4.5. Smoke Test MapReduce

1. Try browsing to the ResourceManager:

```
http://$resourcemanager.full.hostname:8088/
```

2. Smoke test using Terasort and sort 10GB of data.

```
su $HDFS_USER
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-2*.jar teragen 100 /test/10gsort/input
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-2*.jar terasort /test/10gsort/input /test/10gsort/output
```

5. Installing Apache HBase and Apache 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.

5.1. Install the HBase and ZooKeeper RPMs

Prerequisites

1. You must have at least core Hadoop on your system. See Configure the Remote Repositories for more information.

2. Verify the HDP repositories are available:

   ```
   yum list hbase
   ```

   The output should list at least one HBase package similar to the following:

   ```
   hbase.noarch <version>
   ```

   If yum responds with "Error: No matching package to list" as shown below, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. Follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

   ```
   Error: No matching package to list.
   ```

Installation

In a terminal window, type:

- For RHEL/CentOS/Oracle Linux

  ```
  yum install zookeeper hbase
  ```

- For SLES

  ```
  zypper install zookeeper hbase
  ```

- For Ubuntu

  ```
  apt-get install zookeeper hbase
  ```

5.2. 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 bash script files included with the companion files (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`.
   - *$ZOOKEEPER_DATA_DIR* is the directory where ZooKeeper will store data. For example, `/grid/hadoop/zookeeper/data`.
   - *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

3. Initialize the zookeeper data directories with the 'myid' file. Create one file per Zookeeper server, and put the number of that server in each file.

```
vi $ZOOKEEPER_DATA_DIR/myid
```

In the myid file on the first server, enter the corresponding number:

```
1
```

In the myid file on the second server, enter the corresponding number:

```
2
```

In the myid file on the second server, enter the corresponding number:

```
3
```

5.3. 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 to separate temporary directores.

  The files are located in the `configuration_files/hbase` and `configuration_files/zookeeper` directories where you decompressed the companion files.

- 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 `zoo.cfg` and modify the following properties:

     ```
     dataDir=$zk.data.directory.path

     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>$hbase.master.full.hostname</value>
      <description>Enter the HBase Master server hostname</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>
```

3. Edit the `regionservers` file and list all the RegionServers hostnames (separated by newline character) in your environment. For example, see the sample `regionservers` file with hostnames `RegionServer1` through `RegionServer9`.

```
RegionServer1
RegionServer2
RegionServer3
RegionServer4
RegionServer5
RegionServer6
RegionServer7
RegionServer8
RegionServer9
```

- 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`.

5.4. 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 $HDFS_USER>
   /usr/lib/hadoop/bin/hadoop fs -mkdir /apps/hbase
   /usr/lib/hadoop/bin/hadoop fs -chown -R hbase /apps/hbase
   <login as $HBASE_USER>
   /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:

   ```
   su - $HBASE_USER
   hbase shell
   ```

   In the HBase shell, enter the following command:

   ```
   status
   ```

6. 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

6.1. Install the Pig RPMs

On all the hosts where you will execute Pig programs, install the RPMs.

- For RHEL or CentOS:

```
yum install pig
```

- For SLES:

```
zypper install pig
```

- For Ubuntu:

```
apt-get install pig
```

The RPM will install Pig libraries to `/usr/lib/pig`. Pig configuration files are placed in `/usr/lib/pig/conf`.

6.2. Set Up Configuration Files

Use the following instructions to set up configuration files for Pig:

1. Extract the Pig configuration files.

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

2. Copy the configuration files.

   a. On all hosts where Pig will be executed, create the Pig configuration directory:

   ```
   rm -r $PIG_CONF_DIR
   mkdir -p $PIG_CONF_DIR
   ```

   b. Copy all the configuration files to *$PIG_CONF_DIR*.

   c. Set appropriate permissions:

   ```
   chown -R $PIG_USER:$HADOOP_GROUP $PIG_CONF_DIR
   ```

```
chmod -R 755 $PIG_CONF_DIR
```

where:

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

6.3. Validate the Installation

Use the following steps to validate your installation:

1. On the host machine where Pig is installed execute the following commands:

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

2. Create the pig script file `/tmp/id.pig` with the following contents:

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

3. Execute the Pig script:

```
export JAVA_HOME=/usr/java/default
 pig -l /tmp/pig.log /tmp/id.pig
```

7. 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 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. Create Directories on HDFS
5. Validate the Installation

7.1. Install the Hive and HCatalog RPMs

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

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install hive hcatalog
     ```

   - For SLES:

     ```
     zypper install hive hcatalog
     ```

   - For Ubuntu:

     ```
     apt-get install hive hcatalog
     ```

2. Optional - Download and add the database connector JAR.

   By default, Hive uses embedded Derby database for its metastore. However, you can optionally choose to enable remote database (MySQL) for Hive metastore.

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

      ```
      [For RHEL/CENTOS/ORACLE LINUX]
      yum install mysql-connector-java*
      ```

      ```
      [For SLES]
      zypper install mysql-connector-java*
      ```

```
[For UBUNTU]
apt-get install mysql-connector-java*
```

b. After the install, the mysql jar is placed in '/usr/share/java/'. Copy the downloaded JAR file to the `/usr/lib/hive/lib/` directory on your Hive host machine.

c. Verify that the JAR file has appropriate permissions.

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 bash script files included in companion files (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`.

7.3. Set Up the Hive/HCatalog Configuration Files

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

1. Extract the Hive/HCatalog configuration files to a temporary directory.

   The files are located in `configuration_files/hive` where you decompressed the companion files.

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>jdbc:mysql://$mysql.full.hostname:3306/$database.name?
createDatabaseIfNotExist=true</value>
 <description>Enter your JDBC connection string. </description>
</property>
```

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

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

Enter your MySQL credentials from Install MySQL (Optional).

```
<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>
```

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`.

7.4. Create Directories on HDFS

1. Create Hive user home directory on HDFS.

```
Login as $HDFS_USER
```

```
hadoop fs -mkdir -p /user/$HIVE_USER
hadoop fs -chown $HIVE_USER:$HDFS_USER /user/$HIVE_USER
```

2. Create warehouse directory on HDFS.

```
Login as $HDFS_USER
hadoop fs -mkdir -p /apps/hive/warehouse
hadoop fs -chown -R $HIVE_USER:$HDFS_USER /apps/hive
hadoop fs -chmod -R 775 /apps/hive
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HIVE_USER* is the user owning the Hive services. For example, `hive`.

3. Create hive scratch directory on HDFS.

```
Login as $HDFS_USER
hadoop fs -mkdir -p /tmp/scratch
hadoop fs -chown -R $HIVE_USER:$HDFS_USER /tmp/scratch
hadoop fs -chmod -R 777 /tmp/scratch
```

where:

- *$HDFS_USER* is the user owning the HDFS services. For example, `hdfs`.
- *$HIVE_USER* is the user owning the Hive services. For example, `hive`.

7.5. Validate the Installation

Use the following steps to validate your installation:

1. 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 >$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;
```

where:

- *$HIVE_USER* is the user that owns the HIVE services. For example, `hive`.
- *$HIVE_LOG_DIR* is the directory for storing the Hive Server logs. This directory name is a combination of a directory and the *$HIVE_USER*.

8. Installing Apache WebHCat

This section describes installing and testing Apache 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

8.1. Install the WebHCat RPMs

On the WebHCat server machine, install the necessary RPMs.

- For RHEL/CentOS/Oracle Linux:

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

- For SLES:

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

- For Ubuntu:

```
apt-get install hcatalog webhcat-tar-hive webhcat-tar-pig
```

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 set up Pig configuration files :

1. We strongly suggest that you edit and source the bash script files included in the companion files (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:

- *$WEBHCAT_LOG_DIR* is the directory to store the WebHCat logs. For example, `var/log/webhcat`.
- *$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`.

8.3. Modify WebHCat Configuration Files

Use the following instructions to modify the WebHCat config files:

1. Extract the WebHCat configuration files to a temporary directory.

   The files are located in the `configuration_files/webhcat` directory where you decompressed the companion files.

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-env.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>
   ```

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`.

8.4. Set Up HDFS User and Prepare WebHCat Directories On HDFS

1. Set up the WebHCat user.

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

2. Prepare WebHCat directories on HDFS.

```
hdfs dfs -copyFromLocal /usr/share/HDP-webhcat/pig.tar.gz /apps/webhcat/
hdfs dfs -copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /apps/webhcat/
hdfs dfs -copyFromLocal /usr/lib/hadoop-mapreduce/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`.

8.5. 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"}
```

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

Prerequisites

1. You must have at least core Hadoop on your system. See Configure the Remote Repositories for more information.
2. Verify the HDP repositories are available:

   ```
   yum list oozie
   ```

   The output should list at least one Oozie package similar to the following:

   ```
   oozie.noarch <version>
   ```

   If yum responds with "Error: No matching package to list" as shown below, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. Follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

   ```
   Error: No matching package to list.
   ```

Installation

1. On the Oozie server, install the necessary RPMs.
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install oozie oozie-client
     ```

   - For SLES:

     ```
     zypper install oozie oozie-client
     ```

   - For Ubuntu:

     ```
     apt-get install oozie oozie-client
     ```

2. Optional - Enable the Oozie Web Console

- Create a lib extension directory.

```
cd /usr/lib/oozie
mkdir libext
```

- Add the ExtJS library to the Oozie application.
  - For RHEL/CentOS/Oracle Linux:

```
yum install extjs-2.2-1
cp /usr/share/HDP-oozie/ext-2.2.zip libext/
```

  - For SLES:

```
zypper install extjs-2.2-1
cp /usr/share/HDP-oozie/ext-2.2.zip libext/
```

  - For Ubuntu:

```
apt-get install extjs
cp /usr/share/HDP-oozie/ext-2.2.zip libext/
```

- Add LZO JAR files.

```
cp /usr/lib/hadoop/lib/hadoop-lzo-0.5.0.jar libext/
```

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, delete and recreate them. Use the following instructions to set up Oozie configuration files:

1. We strongly suggest that you edit and source the bash script files included in the companion files (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 to a temporary directory.

   The files are located in the `configuration_files/oozie` directory where you decompressed the companion files.

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>

</property>
```

```
<property>
 <name>oozie.service.JPAService.jdbc.driver</name>
 <value>org.apache.derby.jdbc.EmbeddedDriver</value>
</property>
```

```
<property>
 <name>oozie.service.JPAService.jdbc.username</name>
 <value>$OOZIE_DBUSER</value>
</property>
```

```
<property>
 <name>oozie.service.JPAService.jdbc.password</name>
 <value>$OOZIE_DBPASSWD</value>
</property>
```

b. Edit the `oozie-env.sh` and modify the following properties to match the directories created:

```
<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. Run the setup script to prepare the Oozie Server

```
cd /usr/lib/oozie/
bin/oozie-setup.sh prepare-war
```

2. Create the Oozie DB schema

```
cd /usr/lib/oozie/
bin/ooziedb.sh create -sqlfile oozie.sql -run Validate DB Connection
```

3. Start the Oozie server:

```
<login as $oozie_user>
cd /usr/lib/oozie/
/usr/lib/oozie/bin/oozie-start.sh
```

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

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

5. Access the Oozie Server with the Oozie client.

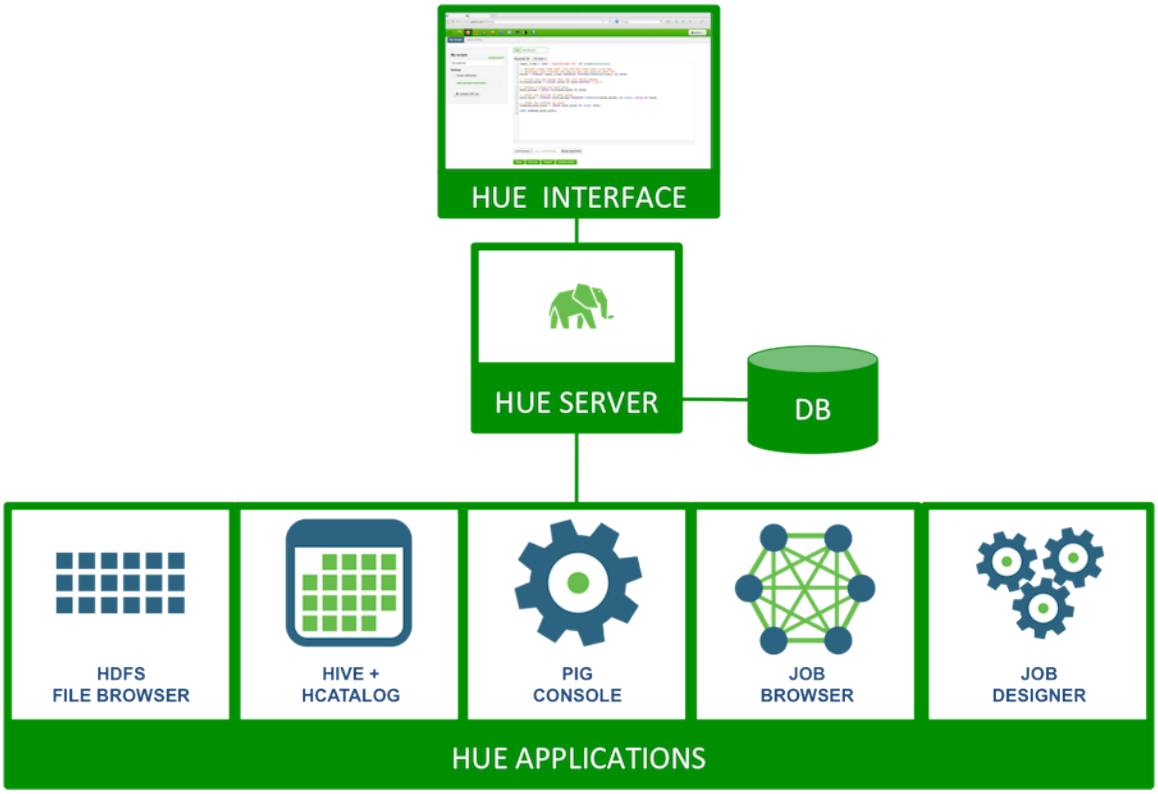
```
oozie admin -oozie http://$oozie.full.hostname:11000/oozie -status
```

You should see the following output:

```
System mode: NORMAL
```

10. Installing Hue

Hue provides a Web application interface for Apache Hadoop. It supports a file browser, JobTracker interface, Hive, Pig, Oozie, HBase, and more.



Complete the following instructions to install Hue:

1. Prerequisites
2. Configure HDP
3. Install Hue
4. Configure Hue
5. Start Hue
6. Validate Hue

10.1. Prerequisites

Complete the following prerequisites before deploying Hue.

1. Verify that you have a host that supports Hue:
   - RHEL, CentOS, Oracle Linux v5 or v6

- Windows (Vista, 7)
- Mac OS X (10.6 or later)

Note

Hue is not supported on Ubuntu.

2. Verify that you have a browser that supports Hue:

Table 10.1. Hue Browser Support

Linux (RHEL, CentOS, Oracle, SLES)	Windows (VISTA, 7)	Mac OS X (10.6 or later)
Firefox latest stable release	Firefox latest stable release	Firefox latest stable release
Google Chrome latest stable release	Google Chrome latest stable release	Google Chrome latest stable release
N/A	Internet Explorer 9 (for Vista + Windows 7)	N/A
N/A	Safari latest stable release	Safari latest stable release

3. For RHEL/CentOs/Oracle 5.x verify that you are deploying the following dependency on all the host machines in your cluster:

```
yum install python26
```

4. Stop all the services in your cluster. For more information see the instructions provided here.

5. Install and run the HDP Hadoop cluster from HDP-2.0.6.0.

   The following table outlines the dependencies on the HDP components:

Table 10.2. Dependencies on the HDP components

Component	Required	Applications	Notes
HDFS	Yes	Core, Filebrowser	HDFS access through WebHDFS or HttpFS
YARN	Yes	JobDesigner, JobBrowser, Beeswax	Transitive dependency via Hive or Oozie
Oozie	No	JobDesigner, Oozie	Oozie access through REST API
Hive	No	Beeswax, HCatalog	Beeswax uses the Hive client libraries
WebHCat	No	HCatalog, Pig	HCatalog and Pig use WebHcat REST API
HBase	No	Shell	Optionally provides access to the HBase shell

6. Choose a Hue Server host machine in your cluster where you want to deploy your Hue Server.

   Typically, you can choose to deploy Hue on any node within your cluster. However, if your corporate firewall policies allow, you can also use a remote host machine as your Hue server. For pilot or small cluster sizes, you can use the master install machine for HDP as your Hue server.

7. Configure the firewall.

   a. Verify that the host machines within your cluster can connect to each other over TCP.

   b. The machines outside your cluster must be able to open TCP port `8000` on the Hue Server (or the configured Hue web HTTP port) to interact with the system.

10.2. Configure HDP

Note

If you are using an Ambari-managed cluster, use Ambari to update the Service configurations (core-site.xml, mapred-site.xml, webhbcat-site.xml and oozie-site.xml). Do not edit the configuration files directly and use Ambari to start and stop the services.

1. Modify the `hdfs-site.xml` file.

   On the NameNode, Secondary NameNode, and all the DataNodes, add the following properties to the *$HADOOP_CONF_DIR*`/hdfs-site.xml` file.

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

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

2. Modify the `core-site.xml` file.

   On the NameNode, Secondary NameNode, and all the DataNodes, add the following properties to the *$HADOOP_CONF_DIR*`/core-site.xml` file.

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

```
<property>
  <name>hadoop.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>hadoop.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

```
<property>
<name>hadoop.proxyuser.hcat.groups</name>
<value>*</value>
</property>
```

```
<property>
<name>hadoop.proxyuser.hcat.hosts</name>
<value>*</value>
</property>
```

3. Modify the `webhcat-site.xml` file. On the WebHCat Server host, add the following properties to the `$WEBHCAT_CONF_DIR/webhcat-site.xml` Where `$WEBHCAT_CONF_DIR` is the directory for storing WebHCat configuration files. For example, `/etc/webhcat/conf`.

```
vi $WEBHCAT_CONF_DIR/webhcat-site.xml
```

```
<property>
  <name>webhcat.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>webhcat.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

4. Modify the `oozie-site.xml` file. On the Oozie Server host, add the following properties to the `$OOZIE_CONF_DIR/oozie-site.xml` Where `$OOZIE_CONF_DIR` is the directory for storing Oozie configuration files. For example, `/etc/oozie/conf`.

```
vi $OOZIE_CONF_DIR/oozie-site.xml
```

```
<property>
  <name>oozie.service.ProxyUserService.proxyuser.hue.hosts</name>
  <value>*</value>
</property>
```

```
<property>
  <name>oozie.service.ProxyUserService.proxyuser.hue.groups</name>
  <value>*</value>
</property>
```

5. Stop the NameNode by running the following command:

```
/usr/lib/hadoop/sbin/hadoop-daemon.sh --config $HADOOP_CONF_DIR stop
 namenode
```

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

10.3. Install Hue

Prerequisites

1. You must have at least core Hadoop on your system. See Configure the Remote Repositories for more information.

2. Verify the HDP repositories are available:

```
yum list hue hue-*
```

   The output should list at least one Hue package similar to the following:

```
hue.x86_64 <version>
hue-beeswax.x86_64 <version>
hue-common.x86_64 <version>
hue-hcatalog.x86_64 <version>
```

```
hue-oozie.x86_64 <version>
hue-pig.x86_64 <version>
hue-server.x86_64 <version>
```

If yum responds with "Error: No matching package to list" as shown below, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. Follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

```
Error: No matching package to list.
```

Installation

Execute the following command on all Hue Server host machines:

- For RHEL/CentOS/Oracle Linux:

```
yum install hue
```

- For SLES:

```
zypper install hue
```

10.4. Configure Hue

Use the following commands to explore the configuration options for Hue.

- To list all available configuration options:

```
/usr/lib/hue/build/env/bin/hue config_help | less
```

- To use multiple files to store your configuration:

  Hue loads and merges all of the files with extension `.ini` located in the `/etc/hue/conf` directory.

Use the following instructions to configure Hadoop for Hue:

1. Configure Web Server
2. Configure Hadoop
3. Configure Beeswax
4. Configure JobDesigner and Oozie
5. Configure UserAdmin
6. Configure WebHCat

10.4.1. Configure Web Server

Use the following instructions to configure Web server:

These configuration variables are under the `[desktop]` section in the `hue.ini` configuration file.

1. Specify the Hue HTTP Address.

   Use the following options to change the IP address and port of the existing Web Server for Hue (by default, Spawning or CherryPy).

   ```
   # Webserver listens on this address and port
   http_host=0.0.0.0
   http_port=8000
   ```

   The default setting is port 8000 on all configured IP addresses.

2. Specify the Secret Key.

   To ensure that your session cookies are secure, enter a series of random characters (30 to 60 characters is recommended) as shown below:

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

3. Configure authentication.

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

4. Configure Hue for SSL.

   Install pyOpenSSL in order to configure Hue to serve over HTTPS. To install pyOpenSSL, from the root of your Hue installation path, complete the following instructions:

   a. Execute the following command on the Hue Server:

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

   b. Configure Hue to use your private key. Add the following to `hue.ini` file:

      ```
      ssl_certificate=$PATH_To_CERTIFICATE
      ssl_private_key=$PATH_To_KEY
      ```

      Ideally, you should have an appropriate key signed by a Certificate Authority. For test purposes, you can create a self-signed key using the `openssl` command on your system:

      ```
       ### Create a key
      openssl genrsa 1024 > host.key
      ```

      ```
      ### Create a self-signed certificate
      openssl req -new -x509 -nodes -sha1 -key host.key > host.cert
      ```

      Note

      To upload files using the Hue File Browser over HTTPS, you must have a proper SSL Certificate.

10.4.2. Configure Hadoop

Use the following instructions to configure Hadoop:

These configuration variables are under the `[hadoop]` section in the `hue.ini` configuration file.

1. Configure HDFS Cluster.

   Hue supports only one HDFS cluster currently.

   Ensure that you define the HDFS cluster under the `[hadoop][[hdfs_clusters]][[[default]]]` sub-section. Use the following variables to configure the HDFS cluster:

fs_defaultfs	This is equivalent to `fs.defaultFS` (`fs.default.name`) in Hadoop configuration. For example, `hdfs://fqdn.namenode.host:8020`
webhdfs_url	You can also set this to be the WebHDFS URL. The default value is the HTTP port on the NameNode. For example, `http://fqdn.namenode.host:50070/webhdfs/v1`
hadoop_hdfs_home	This is the home of your Hadoop HDFS installation. It is the root of the Hadoop untarred directory or usually `/usr/lib/hadoop`.
hadoop_bin	Use this as the HDFS Hadoop launcher script, which is usually `/usr/bin/hadoop`.
hadoop_conf_dir	This is the configuration directory of the HDFS, typically `/etc/hadoop/conf`.

2. Configure YARN (MR2) Cluster.

   Hue supports only one YARN cluster currently.

   Ensure that you define the YARN cluster under the `[hadoop][[yarn_clusters]][[[default]]]` sub-section. Use the following variables to configure the YARN cluster:

resourcemanager_host	The host running the ResourceManager.
resourcemanager_port	The port for the ResourceManager IPC service.
submit_to	Set this property to true. Hue will be submitting jobs to this YARN cluster. But note that JobBrowser will not be able to show MR2 jobs.
hadoop_mapred_home	This is the home of your Hadoop MapReduce installation. It is the root of HDP Hadoop-MapReduce directory (`/usr/lib/hadoop-mapreduce`). If `submit_to` is `true` for this cluster, this configuration value is set as the `$HADOOP_MAPRED_HOME` for BeeswaxServer and child shell processes.
hadoop_bin	Use this as the YARN/MR2 Hadoop launcher script (`/usr/bin/hadoop`).

hadoop_conf_dir	This is the configuration directory of the YARN/MR2 service, typically set to `/etc/hadoop/conf`.
resourcemanager_api_url	The URL of the ResourceManager API. For example, `http://fqdn.resourcemanager.host:8088`.
proxy_api_url	The URL of the ProxyServer API. For example, `http://fqdn.resourcemanager.host:8088`.
history_server_api_url	The URL of the HistoryServer API. For example, `http://fqdn.historyserver.host:19888`.
node_manager_api_url	The URL of the NodeManager API. For example, `http://fqdn.resourcemanager.host:8042`.

10.4.3. Configure Beeswax

In the `[beeswax]` section of the configuration file, you can optionally specify the following:

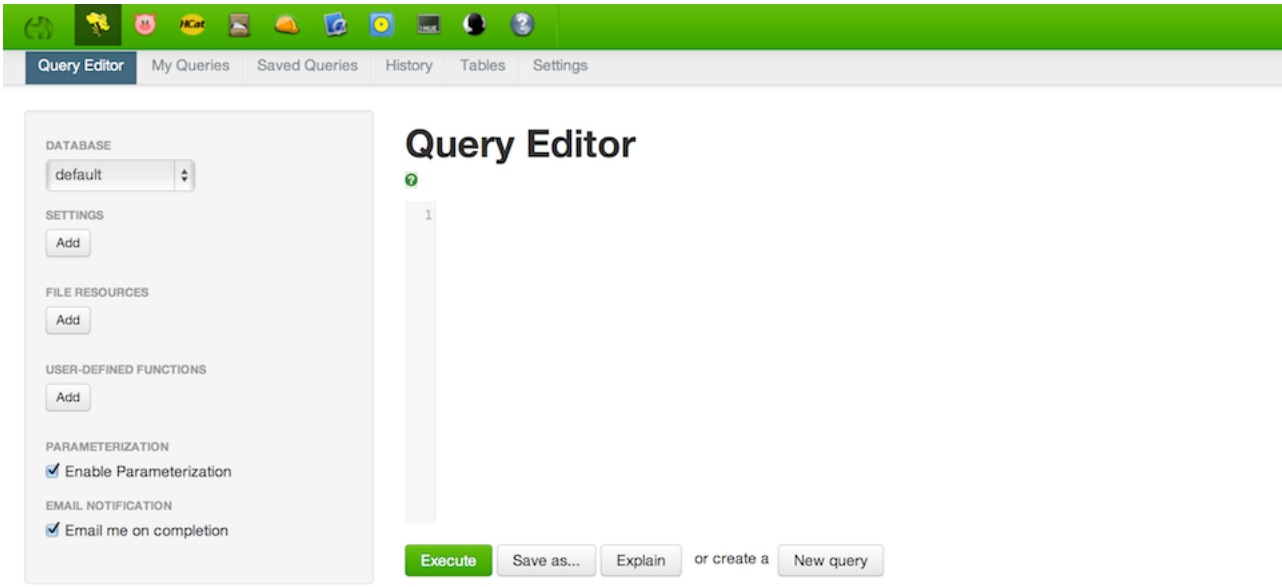
beeswax_server_host	The hostname or IP that the Beeswax Server should bind to. By default it binds to localhost, and therefore only serves local IPC clients.
hive_home_dir	The base directory of your Hive installation.
hive_conf_dir	The directory containing your `hive-site.xml` Hive configuration file.
beeswax_server_heapsize	The heap size (`-Xmx`) of the Beeswax Server.

10.4.3.1. Optional - Configure Beeswax Email Notifications

You can receive email notfications when a query completes.

To configure email notifications:

1. Confirm that the `/etc/hue/conf/hue.ini` file is pointing to the correct SMTP server host and port.
2. Set up your user profile. Select **User Admin** and select your user name for email notifications.
3. Select **Step 2: Names and Groups**.
4. Add your e-mail address and save.
5. From the Beeswax Query Editor, select **Email me on completion** and run your query.



10.4.4. Configure JobDesigner and Oozie

In the `[liboozie]` section of the configuration file, you should specify:

oozie_url — The URL of the Oozie service as specified by the `OOZIE_URL` environment variable for Oozie.

10.4.5. Configure UserAdmin

In the `[useradmin]` section of the configuration file, you can optionally specify:

default_user_group — The name of a default group that is suggested when creating a user manually. If the `LdapBackend` or `PamBackend` are configured for user authentication, new users will automatically be members of the default group.

10.4.6. Configure WebHCat

In the `[HCatalog]` section of the `hue.ini` configuration file, update the following property:

templeton_url — The hostname or IP of the WebHCat server.

10.5. Start Hue

As a root user, execute the following command on the Hue Server:

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

This command starts several subprocesses corresponding to the different Hue components.

Note

To stop Hue, execute the following command:

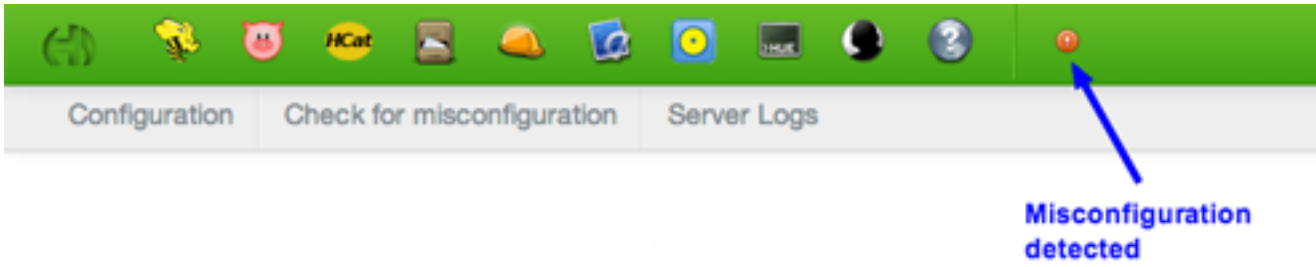
```
/etc/init.d/hue stop
```

To restart Hue, execute the following command:

```
/etc/init.d/hue restart
```

10.6. Validate Configuration

For any invalid configurations, Hue displays a red alert icon on the top navigation bar:



To view the current configuration of your Hue Server, select **About > Configuration** or `http://hue.server:8000/dump_config`.

11. 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 deploy Apache Sqoop:

1. Install the Sqoop RPMs
2. Set Up the Sqoop Configuration
3. Install the Sqoop RPMs

11.1. Install the Sqoop RPMs

Prerequisites

1. You must have at least core Hadoop on your system. See Configure the Remote Repositories for more information.
2. Verify the HDP repositories are available:

   ```
   yum list sqoop
   ```

   The output should list at least one Sqoop package similar to the following:

   ```
   sqoop.noarch <version>
   ```

   If yum responds with "Error: No matching package to list" as shown below, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. Follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

   ```
   Error: No matching package to list.
   ```

Installation

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

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install sqoop
  ```

- For SLES:

  ```
  zypper install sqoop
  ```

- For Ubuntu:

  ```
  apt-get install sqoop
  ```

11.2. Set Up the Sqoop Configuration

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

Use the following instructions to set up Sqoop configuration files:

1. We strongly suggest that you edit and source the bash script files included in the companion files (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. Extract the Sqoop configuration files to a temporary directory.

   The files are located in the `configuration_files/sqoop` directory where you decompressed the companion files.

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. From the file you downloaded in Download Companion Files extract the files in `configuration_files/sqoop` to a temporary directory.

   b. Copy all the config files to the

      ```
      <copy the config files to $SQOOP_CONF_DIR >
      ```

      where *$SQOOP_CONF_DIR* is he directory to store the Sqoop configuration files. For example, `/usr/lib/sqoop/conf`.

11.3. Validate the Installation

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

```
sqoop version | grep 'Sqoop [0-9].*'
```

12. Installing Apache Mahout

Install Apache Mahout on the machine that will run it, either the Hadoop node or your client environment. Do not install it on every node in your cluster.

To install the Mahout RPM use the following command:

- RHEL/CentOS/Oracle Linux:

```
yum install mahout
```

- For SLES:

```
zypper install mahout
```

- For Ubuntu:

```
apt-get install mahout
```

13. Installing and Configuring Apache Flume in HDP

You can manually install and configure Apache Flume to work with the Hortonworks Data Platform (HDP).

Use the following links to install and configure Flume for HDP:

13.1. Understand Flume

Flume is a top-level project at the Apache Software Foundation. While it can function as a general-purpose event queue manager, in the context of Hadoop it is most often used as a log aggregator, collecting log data from many diverse sources and moving them to a centralized data store.

Note

What follows is a very high-level description of the mechanism. For more information, access the Flume HTML documentation set installed with Flume. After you install Flume, access the documentation set at `file:///usr/lib/flume/docs/index.html` on the host on which Flume is installed. The *"Flume User Guide"* is available at `file:///usr/lib/flume/docs/FlumeUserGuide.html`. If you have access to the Internet, the same documentation is also available at the Flume website, flume.apache.org.

13.1.1. Flume Components

A Flume data flow is made up of five main components: Events, Sources, Channels, Sinks, and Agents.

Events — An event is the basic unit of data that is moved using Flume. It is similar to a message in JMS and is generally small. It is made up of headers and a byte-array body.

Sources — The source receives the event from some external entity and stores it in a channel. The source must understand the type of event that is sent to it: an Avro event requires an Avro source.

Channels — A channel is an internal passive store with certain specific characteristics. An in-memory channel, for example, can move events very quickly, but does not provide persistence. A file based channel provides persistence. A source stores an event in the channel where it stays until it is consumed by a sink. This temporary storage lets source and sink run asynchronously.

Sinks — The sink removes the event from the channel and forwards it on either to a destination, like HDFS, or to another agent/dataflow. The sink must output an event that is appropriate to the destination.

Agents — An agent is the container for a Flume data flow. It is any physical JVM running Flume. The same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

13.2. Install Flume

Flume is included in the HDP repository, but it is not installed automatically as part of the standard HDP installation process.

13.2.1. Prerequisites

1. The following Flume components have HDP component dependencies. You cannot use these Flume components if the dependencies are not installed.

   Table 13.1. Flume 1.4.0 Dependencies

Flume 1.4.0 Component	HDP Component Dependencies
HDFS Sink	Hadoop 2.2.0
HBase Sink	HBase 0.96.1

   See HDP Deployment Options for more information.

2. Verify the HDP repositories are available:

   ```
   yum list flume
   ```

   The output should list at least one Flume package similar to the following:

   ```
   flume.noarch 1.5.2.2.2.6.0-2800.el6 HDP-2.2
   ```

   If yum responds with "Error: No matching package to list" as shown below, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. Follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

   ```
   Error: No matching package to list.
   ```

3. You must correctly set and export your JAVA_HOME environment variable for your operating system. See JDK Requirements for instructions on installing JDK.

13.2.2. Installation

Verify the HDP repositories are available for your Flume installation by entering `yum list flume`. See Prerequisites for more information.

To install Flume, from a terminal window type:

- For RHEL or CentOS

```
yum install flume
yum install flume-agent #This installs init scripts
```

- For SLES

```
zypper install flume
zypper install flume-agent #This installs init scripts
```

- For Ubuntu

```
apt-get install flume
apt-get install flume-agent #This installs init scripts
```

13.2.3. Users

The installation process automatically sets up the appropriate `flume` user and `flume` group in the operating system.

13.2.4. Directories

The main Flume files are located in `/usr/lib/flume` and the main configuration files are located in `/etc/flume/conf`.

13.3. Configure Flume

You configure Flume by using a properties file, which is specified on Flume start-up. The init scripts installed by `flume-agent` bring up a single Flume agent on any host, using the contents of `/etc/flume/conf/flume-conf`.

Tip

Hadoop administrators planning to run Flume as a service must assign the name `agent` as the service name for all relevant settings in `flume-conf`.

To see what configuration properties you can adjust, a template for this file is installed in the configuration directory at: `/etc/flume/conf/flume-conf.properties.template`. A second template file exists for setting environment variables automatically at start-up: `/etc/flume/conf/flume-env.sh.template`.

Common configuration option choices include the following:

- Set primary configuration options in `/etc/flume/conf/flume-conf`:
  - If you are using the HDFS sink make sure the target folder is in HDFS

- Set environment options in `/etc/flume/conf/flume-env.sh`:
  - To enable JMX monitoring, add the following properties to JAVA_OPTS

```
JAVA_OPTS="-Dcom.sun.management.jmxremote
-Dcom.sun.management.jmxremote.port=4159
-Dcom.sun.management.jmxremote.authenticate=false
-Dcom.sun.management.jmxremote.ssl=false"
```

  - To enable Ganglia monitoring, add the following properties to JAVA_OPTS

```
JAVA_OPTS="-Dflume.monitoring.type=ganglia
-Dflume.monitoring.hosts=<ganglia-server>:8660"
```

    Where <ganglia-server> is the name of the Ganglia server host.

  - To optimize the heap size, add the following properties to JAVA_OPTS

```
JAVA_OPTS= "-Xms100m -Xmx200m"
```

- Set the log directory for log4j in `/etc/flume/conf/log4j.properties`

```
flume.log.dir=/var/log/flume
```

13.4. Start Flume

There are two options for starting Flume.

- Start Flume directly. On the Flume host:

```
/etc/rc.d/init.d/flume-agent start
```

- Start Flume as a service. On the Flume host:

```
service flume-agent start
```

Tip

Hadoop administrators planning to run Flume as a service must assign the name `agent` as the service name for all relevant configuration settings in `flume-conf`.

13.5. HDP and Flume

Flume ships with many source, channel, and sink types. For use with HDP the following types have been thoroughly tested:

13.5.1. Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp

13.5.2. Channels

- Memory
- File

13.5.3. Sinks

- HDFS: secure, nonsecure
- HBase

13.6. A Simple Example

The following snippet shows some of the kinds of properties that can be set using the properties file. For more detailed information, see the *"Flume User Guide"*.

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory
agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd
agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an `exec` source, the agent runs a given command on start-up which streams data to `stdout`, where the source gets it. In this case, the command is a Python test script. The channel is defined as an in-memory channel and the sink is an HDFS sink.

14. Installing Ganglia

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

14.1. Install the Ganglia RPMs

On the host you have chosen to be the Ganglia server, install the server RPMs.

- For RHEL/CentOS/Oracle Linux:

```
yum install ganglia-gmond-3.5.0-99 ganglia-gmetad-3.5.0-99 ganglia-web-3.5.7-99
```

- For SLES:

```
zypper install ganglia-gmond-3.5.0-99 ganglia-gmetad-3.5.0-99 ganglia-web-3.5.7-99
```

On each host in the cluster, install the client RPMs:

- For RHEL/CentOS/Oracle Linux:

```
yum install ganglia-gmond-3.5.0-99
```

- For SLES:

```
zypper install ganglia-gmond-3.5.0-99
```

14.2. Install the Configuration Files

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

14.2.1. Extract the Ganglia Configuration Files

From the file you downloaded in Download Companion Files, open the `configuration_files.zip` and copy the files in the `ganglia` folder to a temporary directory. The `ganglia` folder contains two sub-folders, `objects` and `scripts`.

14.2.2. Copy the Configuration Files

On each host in the cluster:

1. Create the directory for the `objects` folder:

```
mkdir -p /usr/libexec/hdp/ganglia
```

2. Copy the objects files:

```
cp <tmp-directory>/ganglia/objects/*.* /usr/libexec/hdp/ganglia/
```

3. Copy the Ganglia monitoring init script to `init.d`

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

On the Ganglia Server Host :

1. Copy the entire contents of the scripts folder to `init.d`

```
cp -R <tmp-directory>/ganglia/scripts/* /etc/init.d/
```

14.2.3. Set Up Ganglia Hosts

1. On the Ganglia server, to configure the `gmond` collector:

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

2. If HBase is installed, on the HBase Master:

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

3. On the NameNode and SecondaryNameNode servers, to configure the `gmond` emitters:

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

4. On the ResourceManager server, to configure the `gmond` emitters:

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

5. On all hosts, to configure the `gmond` emitters:

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

6. If HBase is installed, on the HBase Master, to configure the `gmond` emitter:

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

14.2.4. Set Up Configurations

1. On the Ganglia server, use a text editor to open the following master configuration files:

```
/etc/ganglia/hdp/HDPNameNode/conf.d/gmond.master.conf
/etc/ganglia/hdp/HDPHistoryServer/conf.d/gmond.master.conf
/etc/ganglia/hdp/HDPResourceManager/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
```

2. Confirm that the “bind” property in each of these files is set to the Ganglia server hostname.

3. On the Ganglia server, use a text editor to open the `gmetad` configuration file:

   `/etc/ganglia/hdp/gmetad.conf`

4. Confirm 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 "HDPResourceManager" my.ganglia.server.hostname:8664
data_source "HDPHistoryServer" my.ganglia.server.hostname:8666
```

And if HBase is installed:

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

5. On all hosts except the Ganglia server, use a text editor to open the slave configuration files:

```
/etc/ganglia/hdp/HDPNameNode/conf.d/gmond.slave.conf
/etc/ganglia/hdp/HDPHistoryServer/conf.d/gmond.slave.conf
/etc/ganglia/hdp/HDPResourceManager/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. Confirm that the `host` property is set to the Ganglia Server hostname.

14.2.5. Set Up Hadoop Metrics

On each host in the cluster:

1. Stop the Hadoop services.
2. Change to the Hadoop configuration directory.

```
cd $HADOOP_CONF_DIR
```

3. Copy the Ganglia metrics properties file into place.

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

4. 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
resourcemanager.sink.ganglia.servers=my.ganglia.server.hostname:8664
nodemanager.sink.ganglia.servers=my.ganglia.server.hostname:8660
historyserver.sink.ganglia.servers=my.ganglia.server.hostname:8666
maptask.sink.ganglia.servers=my.ganglia.server.hostname:8660
reducetask.sink.ganglia.servers=my.ganglia.server.hostname:8660
```

5. Restart the Hadoop services.

14.3. Validate the Installation

Use these steps to validate your installation.

14.3.1. Start the Ganglia Server

On the Ganglia server:

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

14.3.2. Start Ganglia Monitoring on All Hosts

On all hosts:

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

14.3.3. Confirm that Ganglia is Running

Browse to the Ganglia server:

```
http://{ganglia.server}/ganglia
```

15. Installing Nagios

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

15.1. Install the Nagios RPMs

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

```
[For RHEL and CentOS]
yum -y install net-snmp net-snmp-utils php-pecl-json
yum -y install wget httpd php net-snmp-perl perl-Net-SNMP fping nagios nagios-
plugins hdp_mon_nagios_addons nagios-www
```

```
[For SLES]
zypper -n --no-gpg-checks install net-snmp
zypper -n --no-gpg-checks install wget apache2 php php-curl perl-SNMP perl-
Net-SNMP fping nagios nagios-plugins hdp_mon_nagios_addons nagios-www
```

15.2. Install the Configuration Files

There are several configuration files that must be set up for Nagios.

15.2.1. Extract the Nagios Configuration Files

From the file you downloaded in Download Companion Files, open the `configuration_files.zip` and copy the files in the `nagios` folder to a temporary directory. The `nagios` folder contains two sub-folders, `objects` and `plugins`.

15.2.2. Create the Nagios Directories

1. Make the following Nagios directories:

```
mkdir /var/nagios /var/nagios/rw /var/log/nagios /var/log/nagios/spool/
checkresults /var/run/nagios
```

2. Change ownership on those directories to the Nagios user:

```
chown -R nagios:nagios /var/nagios /var/nagios/rw /var/log/nagios /var/log/
nagios/spool/checkresults /var/run/nagios
```

15.2.3. Copy the Configuration Files

1. Copy the contents of the `objects` folder into place:

```
cp <tmp-directory>/nagios/objects/*.* /etc/nagios/objects/
```

2. Copy the contents of the `plugins` folder into place:

```
cp <tmp-directory>/nagios/plugins/*.* /usr/lib64/nagios/plugins/
```

15.2.4. Set the Nagios Admin Password

1. Choose a Nagios administrator password, for example, “admin”.
2. Set the password. Use the following command:

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

15.2.5. Set the Nagios Admin Email Contact Address

1. Open `/etc/nagios/objects/contacts.cfg` with a text editor.
2. Change the `nagios@localhost` value to the admin email address so it can recceive alerts.

15.2.6. Register the Hadoop Configuration Files

1. Open `/etc/nagios/nagios.cfg` with a text editor.
2. In the section `OBJECT CONFIGURATION FILE(S)`, add the following:

```
# 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
```

3. Change the `command-file` directive to `/var/nagios/rw/nagios.cmd`:

```
command_file=/var/nagios/rw/nagios.cmd
```

15.2.7. Set Hosts

1. Open `/etc/nagios/objects/hadoop-hosts.cfg` with a text editor.
2. Create 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
  }
```

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

15.2.8. Set Host Groups

1. Open `/etc/nagios/objects/hadoop-hostgroups.cfg` with a text editor.
2. 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 15.1. 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

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

Table 15.2. 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

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

Table 15.3. 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

Service	Component	Name	Alias	Members
WebHCat		webhcat-server	webhcat-server	The WebHCat server
Templeton		templeton-server	templeton-server	The Templeton server

15.2.9. Set Services

1. 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, Templeton and Oozie

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

3. Replace the parameter `@NAGIOS_BIN@` and `@STATUS_DAT@` parameters based on the operating system.

```
[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
```

```
[For Ubuntu]
@STATUS_DAT@ = /var/lib/nagios/status.dat
@NAGIOS_BIN@ = /usr/sbin/nagios
```

4. 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.`

15.2.10. Set Status

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

2. Replace the @STATUS_DAT@ parameter with the location of the Nagios status file. The file is located:

```
[For RHEL and CentOS]
/var/nagios/status.dat
```

```
[For SLES]
/var/lib/nagios/status.dat
```

```
[For Ubuntu]
/var/cache/nagios3/status.dat
```

15.3. Validate the Installation

Use these steps to validate your installation.

15.3.1. Validate the Nagios Installation

Validate the installation.

```
nagios -v /etc/nagios/nagios.cfg
```

15.3.2. Start Nagios and httpd

Start the Nagios server and httpd.

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

15.3.3. Confirm Nagios is Running

Confirm the server is running.

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

This should return:

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

15.3.4. Test Nagios Services

Run 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#>
```

15.3.5. Test Nagios Access

1. Browse to the Nagios server:

   ```
   http://<nagios.server>/nagios
   ```

2. Login using the Nagios admin username (nagiosadmin) and password (see Set the Nagios Admin Password).

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

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

15.3.6. Test Nagios Alerts

1. Login to one of your cluster DataNodes.

2. Stop the TaskTracker service.

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

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

4. Start the TaskTracker service.

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

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

16. Manual Install Appendix: Tarballs

Individual links to the Apache structured tarball files for the projects included with Hortonworks Data Platform are listed below:

- **RHEL 5 and CentOS 5:**

Table 16.1. RHEL/CentOS 5

Project	Download
Hadoop	hadoop-2.2.0.2.0.6.0-102.tar.gz
Pig	pig-0.12.0.2.0.6.1-102.tar.gz
Hive and HCatalog	hive-0.12.0.2.0.6.1-102.tar.gz
HBase and ZooKeeper	hbase-0.96.1.2.0.6.1-102-hadoop2-bin.tar.gz zookeeper-3.4.5.2.0.6.0-102.tar.gz
Oozie	oozie-4.0.0.2.0.6.0-102-distro.tar.gz
Sqoop	sqoop-1.4.4.2.0.6.1-102.bin__hadoop-2.2.0.2.0.6.0-102.tar.gz
Flume	apache-flume-1.4.0.2.0.6.1-102-bin.tar.gz
Mahout	mahout-distribution-0.8.0.2.0.6.1-102.tar.gz

- **RHEL 6 and CentOS 6:**

Table 16.2. RHEL/CentOS 6

Project	Download
Hadoop	hadoop-2.2.0.2.0.6.0-102.tar.gz
Pig	pig-0.12.0.2.0.6.1-102.tar.gz
Hive and HCatalog	hive-0.12.0.2.0.6.1-102.tar.gz
HBase and ZooKeeper	hbase-0.96.1.2.0.6.1-102-hadoop2-bin.tar.gz zookeeper-3.4.5.2.0.6.0-102.tar.gz
Oozie	oozie-4.0.0.2.0.6.0-102-distro.tar.gz
Sqoop	sqoop-1.4.4.2.0.6.1-102.bin__hadoop-2.2.0.2.0.6.0-102.tar.gz
Flume	apache-flume-1.4.0.2.0.6.1-102-bin.tar.gz
Mahout	mahout-distribution-0.8.0.2.0.6.1-102.tar.gz

- **SLES 11:**

Table 16.3. SLES 11

Project	Download
Hadoop	hadoop-2.2.0.2.0.6.0-102.tar.gz
Pig	pig-0.12.0.2.0.6.1-102.tar.gz
Hive and HCatalog	hive-0.12.0.2.0.6.1-102.tar.gz
HBase and ZooKeeper	hbase-0.96.1.2.0.6.1-102-hadoop2-bin.tar.gz zookeeper-3.4.5.2.0.6.0-102.tar.gz
Oozie	oozie-4.0.0.2.0.6.0-102-distro.tar.gz
Sqoop	sqoop-1.4.4.2.0.6.1-102.bin__hadoop-2.2.0.2.0.6.0-102.tar.gz

Project	Download
Flume	apache-flume-1.4.0.2.0.6.1-102-bin.tar.gz
Mahout	mahout-distribution-0.8.0.2.0.6.1-102.tar.gz

- **Ubuntu 12.04:**

Table 16.4. Ubuntu 12.04

Project	Download
Hadoop	hadoop-2.2.0.2.0.6.0-102.tar.gz
Pig	pig-0.12.0.2.0.6.1-102.tar.gz
Hive and HCatalog	hive-0.12.0.2.0.6.1-102.tar.gz
HBase and ZooKeeper	hbase-0.96.1.2.0.6.1-102-hadoop2-bin.tar.gz zookeeper-3.4.5.2.0.6.0-102.tar.gz
Oozie	oozie-4.0.0.2.0.6.0-102-distro.tar.gz
Sqoop	sqoop-1.4.4.2.0.6.1-102.bin__hadoop-2.2.0.2.0.6.0-102.tar.gz
Flume	apache-flume-1.4.0.2.0.6.1-102-bin.tar.gz
Mahout	mahout-distribution-0.8.0.2.0.6.1-102.tar.gz

17. Upgrade HDP Manually

This document provides instructions on how to upgrade to HDP 2.0 from the HDP 1.3 release. Use the following instructions to upgrade to the latest release of HDP:

1. Getting Ready to Upgrade
2. Upgrade Hadoop
3. Migrate the HDP Configurations
4. Create Local Directories
5. Start HDFS
6. Upgrade Apache ZooKeeper
7. Upgrade Apache HBase
8. Upgrade Apache Hive and Apache HCatalog
9. Upgrade Apache Oozie
10. Upgrade Apache WebHCat (Templeton)
11. Upgrade Apache Pig
12. Upgrade Apache Sqoop
13. Upgrade Apache Flume
14. Upgrade vMahout
15. Upgrade Hue

17.1. Getting Ready to Upgrade

HDP Stack upgrade involves removing HDP 1.x MapReduce and replacing it with HDP 2.x Yarn and MapReduce2. Before you begin, review the upgrade process and complete the Backup steps.

1. Back up the following HDP 1.x directories:
   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hcatalog/conf`
   - `/etc/hive/conf`

- /etc/pig/conf
- /etc/sqoop/conf
- /etc/flume/conf
- /etc/mahout/conf
- /etc/oozie/conf
- /etc/hue/conf
- /etc/zookeeper/conf
- Optional - Back up your userlogs directories, ${mapred.local.dir}/userlogs.

2. Run the fsck command as the HDFS Service user and fix any errors. (The resulting file contains a complete block map of the file system.)

```
su $HDFS_USER
hadoop fsck / -files -blocks -locations > /tmp/dfs-old-fsck-1.log
```

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

3. Use the following instructions to compare status before and after the upgrade:

Note

The following commands must be executed by the user running the HDFS service (by default, the user is hdfs).

a. Capture the complete namespace of the file system. (The following command does a recursive listing of the root file system.)

```
su $HDFS_USER
hadoop dfs -lsr / > dfs-old-lsr-1.log
```

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

b. Run the report command to create a list of DataNodes in the cluster.

```
su $HDFS_USER
hadoop dfsadmin -report > dfs-old-report-1.log
```

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

c. Optional - You can copy all or unrecoverable only data stored in HDFS to a local file system or to a backup instance of HDFS.

d. Optional - You can also repeat the steps 3 (a) through 3 (c) and compare the results with the previous run to ensure the state of the file system remained unchanged.

4. As the HDFS user, save the namespace by executing the following command:

```
su $HDFS_USER
hadoop dfsadmin -safemode enter
```

```
hadoop dfsadmin -saveNamespace
```

5. Backup your NameNode metadata.

   a. Copy the following checkpoint files into a backup directory:

      - `dfs.name.dir/edits`
      - `dfs.name.dir/image/fsimage`
      - `dfs.name.dir/current/fsimage`

   b. Store the `layoutVersion` of the namenode.

      `${dfs.name.dir}/current/VERSION`

6. Finalize the state of the filesystem.

```
su $HDFS_USER
hadoop namenode -finalize
```

7. Optional - Backup the Hive Metastore database.

Note

These instructions are provided for your convenience. Please check your database documentation for the latest back up instructions.

Table 17.1. Hive Metastore Database Backup and Rstore

Database Type	Backup	Restore
MySQL	`mysqldump $dbname > $outputfilename.sql` For example: `mysqldump hive > /tmp/mydir/backup_hive.sql`	`mysql $dbname < $inputfilename.sql` For example: `mysql hive < /tmp/mydir/backup_hive.sql`
Postgres	`sudo -u $username pg_dump $databasename > $outputfilename.sql` For example: `sudo -u postgres pg_dump hive > /tmp/mydir/backup_hive.sql`	`sudo -u $username psql $databasename < $inputfilename.sql` For example: `sudo -u postgres psql hive < /tmp/mydir/backup_hive.sql`
Oracle	Connect to the Oracle database using `sqlplus` export the database: `exp username/password@database full=yes file=output_file.dmp`	Import the database: `imp username/password@database ile=input_file.dmp`

8. Optional - Backup the Oozie Metastore database.

Note

These instructions are provided for your convenience. Please check your database documentation for the latest back up instructions.

Table 17.2. Oozie Metastore Database Backup and Restore

Database Type	Backup	Restore
MySQL	`mysqldump $dbname > $outputfilename.sql` For example: `mysqldump oozie > /tmp/mydir/backup_oozie.sql`	`mysql $dbname < $inputfilename.sql` For example: `mysql oozie < /tmp/mydir/backup_oozie.sql`
Postgres	`sudo -u $username pg_dump $databasename > $outputfilename.sql` For example: `sudo -u postgres pg_dump oozie > /tmp/mydir/backup_oozie.sql`	`sudo -u $username psql $databasename < $inputfilename.sql` For example: `sudo -u postgres psql oozie < /tmp/mydir/backup_oozie.sql`

9. Stop all services (including MapReduce) and client applications deployed on HDFS using the instructions provided here.

10. Verify that edit logs in `${dfs.name.dir}/name/current/edits*` are empty. These log files should have only 4 bytes of data, which contain the edit logs version. If the edit logs are not empty, start the existing version NameNode and then shut it down after a new fsimage has been written to disks so that the edit log becomes empty.

17.2. Upgrade Hadoop

1. On all nodes, clean the yum repository.

   - For RHEL/CentOS:

     ```
     cd /etc/yum.repos.d
     yum clean all
     ```

   - For SLES:

     ```
     cd /etc/zypp/repos.d
     zypper clean --all
     ```

2. Uninstall the HDP 1.x packages.

   - For RHEL/CentOS:

     ```
     yum erase hadoop-pipes hadoop-sbin hadoop-native oozie
     ```

   - For SLES:

     ```
     zypper rm hadoop-pipes hadoop-sbin hadoop-native oozie hbase hadoop*
     ```

Note

If you are upgrading from HDP 2.0.6.0 to HDP 2.0.6.1, remove Oozie from the command:

- For RHEL/CentOS:

  ```
  yum erase hadoop-pipes hadoop-sbin hadoop-native
  ```

- For SLES:

```
zypper rm hadoop-pipes hadoop-sbin hadoop-native hbase hadoop*
```

3. Configure your 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.

   a. For each node in your cluster, download the yum repo configuration file `hdp.repo`. From a terminal window, enter the following wget command.

      - For RHEL/CentOS/Oracle Linux 5 :

      ```
      wget  http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.0.6.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
      ```

      - For RHEL/CentOS/Oracle Linux 6:

      ```
      wget  http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.0.6.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
      ```

      - For SLES 11:

      ```
      wget  http://public-repo-1.hortonworks.com/HDP/suse11/2.x/updates/2.0.6.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
      ```

      - For Ubuntu:

      ```
      wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/hdp.list -O /etc/apt-get/repos.d/hdp.list
      ```

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

      - For RHEL/CentOS/Oracle Linux:

```
yum repolist
```

- For SLES:

```
zypper repos
```

- For Ubuntu:

```
apt-get list
```

4. Install Hadoop

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade hadoop*
```

- For SLES:

```
zypper install hadoop* hadoop-hdfs hadoop-lzo
```

- For Ubuntu:

```
apt-get update
apt-get install hadoop hadoop-hdfs libhdfs0 libhdfs0-dev hadoop-yarn
 hadoop-mapreduce hadoop-client opensslliblzo2-2 liblzo2-dev hadoop-lzo
```

5. Install YARN

- For RHEL/CentOS/Oracle Linux:

```
yum install hadoop-mapreduce hadoop-yarn
```

- For SLES:

```
zypper install hadoop-mapreduce hadoop-yarn
```

- For Ubuntu:

```
apt-get install hadoop-mapreduce hadoop-yarn
```

6. Verify HDP 2.x packages have installed successfully.

- For RHEL/CentOS/Oracle Linux:

```
yum list hadoop*|grep HDP-2
```

- For SLES:

```
zypper pa|grep HDP-2
```

Verify that you have HDP 2.x installed:

```
hadoop version
```

You may need to add /etc/hadoop/conf/hadoop-env.sh in /usr/bin/hadoop for $JAVA_HOME.

- For Ubuntu:

```
dpkg -s HDP-2 | grep Status
```

17.3. Migrate the HDP Configurations

Configurations and configuration file names have changed between HDP 1.3.2 (Hadoop 1.2.x) and HDP 2.0.6 (Hadoop 2.1.0-Beta). To successfully upgrade to HDP 2.x, back up your current configuration files, download the new HDP 2 files, and compare. The following tables provide mapping information to make the comparison between releases easier.

To migrate the HDP Configurations

1. Back up the following HDP 1.x configurations on all nodes in your clusters.
   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hcatalog/conf`
   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/zookeeper/conf`
2. Download the your HDP 2.x companion files from Download Companion Files and migrate your HDP 1.x configuration.
3. Copy `log4j.properties` from the hadoop config directory of the companion files to `/etc/hadoop/conf`. The file should have owners and permissions similar to other files in `/etc/hadoop/conf`.
4. Copy these configurations to all nodes in your clusters.
   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hcatalog/conf`

- `/etc/hive/conf`
- `/etc/pig/conf`
- `/etc/sqoop/conf`
- `/etc/flume/conf`
- `/etc/mahout/conf`
- `/etc/oozie/conf`
- `/etc/zookeeper/conf`

Note

Upgrading the repo using yum or zypper resets all configurations. Prepare to replace these configuration directories each time you perform a yum or zypper upgrade.

5. Review the following HDP 1.3.2 Hadoop Core configurations and the new configurations or locations in HDP 2.x

Table 17.3. HDP 1.3.2 Hadoop Core Site (core-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
fs.default.name	core-site.xml	fs.defaultFS	core-site.xml
fs.checkpoint.dir	core-site.xml	dfs.namenode.checkpoint.dir	hdfs-site.xml
fs.checkpoint.edits.dir	core-site.xml	dfs.namenode.checkpoint.edits.dir	hdfs-site.xml
fs.checkpoint.period	core-site.xml	dfs.namenode.checkpoint.period	hdfs-site.xml
io.bytes.per.checksum	core-site.xml	dfs.bytes-per-checksum	hdfs-site.xml
io.bytes.per.checksum	core-site.xml	dfs.bytes-per-checksum	hdfs-site.xml
dfs.df.interval	hdfs-site	fs.df.interval	core-site.xml
hadoop.native.lib	core-site.xml	io.native.lib.available	core-site.xml
hadoop.configured.node.mapping		net.topology.configured.node.mapping	core-site.xml
topology.node.switch.mapping.impl	core-site.xml	net.topology.node.switch.mapping.impl	core-site.xml
topology.script.file.name	core-site.xml	net.topology.script.file.name	core-site.xml
topology.script.number.args	core-site.xml	net.topology.script.number.args	core-site.xml

6. Review the following 1.3.2 HDFS site configations and their new configurations and files in HDP 2.x.

Table 17.4. HDP 1.3.2 Hadoop Core Site (hdfs-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
dfs.block.size	hdfs-site.xml	dfs.blocksize	hdfs-site.xml
dfs.write.packet.size	hdfs-site.xml	dfs.client-write-packet-size	hdfs-site.xml
dfs.https.client.keystore.resource	hdfs-site.xml	dfs.client.https.keystore.resource	hdfs-site.xml
dfs.https.need.client.auth	hdfs-site.xml	dfs.client.https.need-auth	hdfs-site.xml
dfs.read.prefetch.size	hdfs-site.xml	dfs.bytes-per-checksum	hdfs-site.xml

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
dfs.socket.timeout	hdfs-site.xml	dfs.client.socket-timeout	hdfs-site.xml
dfs.balance.bandwidthPerSec	hdfs-site.xml	dfs.datanode.balance.bandwidthPerSec	hdfs-site.xml
dfs.data.dir	hdfs-site.xml	dfs.datanode.data.dir	hdfs-site.xml
dfs.datanode.max.xcievers	hdfs-site.xml	dfs.datanode.max.transfer.threads	hdfs-site.xml
session.id	hdfs-site.xml	dfs.metrics.session-id	hdfs-site.xml
dfs.access.time.precision	hdfs-site.xml	dfs.namenode.accesstime.precision	hdfs-site.xml
dfs.backup.address	hdfs-site.xml	dfs.namenode.backup.address	hdfs-site.xml
dfs.backup.http.address	hdfs-site.xml	dfs.namenode.backup.http-address	hdfs-site.xml
fs.checkpoint.dir	hdfs-site.xml	dfs.namenode.checkpoint.dir	hdfs-site.xml
fs.checkpoint.edits.dir	hdfs-site.xml	dfs.namenode.checkpoint.edits.dir	hdfs-site.xml
fs.checkpoint.period	hdfs-site.xml	dfs.namenode.checkpoint.period	hdfs-site.xml
dfs.name.edits.dir	hdfs-site.xml	dfs.namenode.backup.address	hdfs-site.xml
heartbeat.recheck.interval	hdfs-site.xml	dfs.namenode.heartbeat.recheck-interval	hdfs-site.xml
dfs.http.address	hdfs-site.xml	dfs.namenode.http-address	hdfs-site.xml
dfs.https.address	hdfs-site.xml	dfs.namenode.https-address	hdfs-site.xml
dfs.max.objects	hdfs-site.xml	dfs.namenode.max.objects	hdfs-site.xml
dfs.name.dir	hdfs-site.xml	dfs.namenode.name.dir	hdfs-site.xml
dfs.name.dir.restore	hdfs-site.xml	dfs.namenode.name.dir.restore	hdfs-site.xml
dfs.replication.considerLoad	hdfs-site.xml	dfs.namenode.replication.considerLoad	hdfs-site.xml
dfs.replication.interval	hdfs-site.xml	dfs.namenode.replication.interval	hdfs-site.xml
dfs.max-repl-streams	hdfs-site.xml	dfs.namenode.replication.max-streams	hdfs-site.xml
dfs.replication.min	hdfs-site.xml	dfs.namenode.replication.min	hdfs-site.xml
dfs.replication.pending.timeout-sec	hdfs-site.xml	dfs.namenode.replication.pending.timeout-sec	hdfs-site.xml
dfs.safemode.extension	hdfs-site.xml	dfs.namenode.safemode.extension	hdfs-site.xml
dfs.safemode.threshold.pct	hdfs-site.xml	dfs.namenode.safemode.threshold-pct	hdfs-site.xml
dfs.secondary.http.address	hdfs-site.xml	dfs.namenode.secondary.http-address	hdfs-site.xml
dfs.permissions	hdfs-site.xml	dfs.permissions.enabled	hdfs-site.xml
dfs.permissions.supergroup	hdfs-site.xml	dfs.permissions.superusergroup	hdfs-site.xml
dfs.df.interval	hdfs-site.xml	fs.df.interval	core-site.xml
dfs.umaskmode	hdfs-site.xml	fs.permissions.umask-mode	hdfs-site.xml

7. Review the follwing HDP 1.3.2 MapReduce Configs and their new HDP 2.x Mappings

Table 17.5. HDP 1.3.2 Configs now in Capacity Scheduler for HDP 2.x (mapred-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
mapred.map.child.java.opts	mapred-site.xml	mapreduce.map.java.opts	mapred-site.xml
mapred.job.map.memory.mb	mapred-site.xml	mapreduce.map.memory.mb	mapred-site.xml
mapred.reduce.child.java.opts	mapred-site.xml	mapreduce.reduce.java.opts	mapred-site.xml

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
mapred.job.reduce.memory.mb	mapred-site.xml	mapreduce.reduce.memory.mb	mapred-site.xml

8. Review the following HDP 1.3.2 Configs and their new HDP 2.x Capacity Scheduler mappings.

Table 17.6. HDP 1.3.2 Configs now in Capacity Scheduler for HDP 2.x (capacity-scheduler.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.0.6 config	HDP 2.0.6 config file
mapred.queue.names	mapred-site.xml	yarn.scheduler.capacity.root.queues	capacity-scheduler.xml
mapred.queue.default.acl-submit-job	mapred-queue-acls.xml	yarn.scheduler.capacity.root.default.acl_submit_jobs	capacity-scheduler.xml
mapred.queue.default.acl-administer-jobs	mapred-queue-acls.xml	yarn.scheduler.capacity.root.default.acl_administer_jobs	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.capacity	capacity-scheduler.xml	yarn.scheduler.capacity.root.default.capacity	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.user-limit-factor	capacity-scheduler.xml	yarn.scheduler.capacity.root.default.user-limit-factor	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.maximum-capacity	capacity-scheduler.xml	yarn.scheduler.capacity.root.default.maximum-capacity	capacity-scheduler.xml
mapred.queue.default.state	capacity-scheduler.xml	yarn.scheduler.capacity.root.default.state	capacity-scheduler.xml

9. Compare the following HDP 1.3.2 configs in hadoop-env.sh with the new configs in HDP 2.x

Table 17.7. HDP 1.3.2 Configs and HDP 2.x for hadoop-env.sh

HDP 1.3.2 config	HDP 2.0.6 config	Description
JAVA_HOME	JAVA_HOME	Java implementation to use
HADOOP_HOME_WARN_SUPPRESS	HADOOP_HOME_WARN_SUPPRESS	
HADOOP_CONF_DIR	HADOOP_CONF_DIR	Hadoop Configuration Directory
Not in hadoop-env.sh.	HADOOP_HOME	
Not in hadoop-env.sh.	HADOOP_LIBEXEC_DIR	
HADOOP_NAMENODE_INIT_HEAPSIZE	HADOOP_NAMENODE_INIT_HEAPSIZE	
HADOOP_OPTS	HADOOP_OPTS	Extra Java runtime options. Empty by default.
HADOOP_NAMENODE_OPTS	HADOOP_NAMENODE_OPTS	Command specific options appended to HADOOP_OPTS.
HADOOP_JOBTRACKER_OPTS	Not in hadoop-env.sh.	Command specific options appended to HADOOP_OPTS.
HADOOP_TASKTRACKER_OPTS	Not in hadoop-env.sh.	Command specific options appended to HADOOP_OPTS.
HADOOP_DATANODE_OPTS	HADOOP_DATANODE_OPTS	Command specific options appended to HADOOP_OPTS.
Not in hadoop-env.sh.	YARN_RESOURCEMANAGER_OPTS	Command specific options appended to HADOOP_OPTS.
HADOOP_BALANCER_OPTS	HADOOP_BALANCER_OPTS	Command specific options appended to HADOOP_OPTS.

HDP 1.3.2 config	HDP 2.0.6 config	Description
HADOOP_SECONDARYNAMENODE_OPTS	HADOOP_SECONDARYNAMENODE_OPTS	Command specific options appended to HADOOP_OPTS.
HADOOP_CLIENT_OPTS	HADOOP_CLIENT_OPTS	Applies to multiple commands (fs, dfs, fsck, distcp etc).
HADOOP_SECURE_DN_USER	Not in hadoop-env.sh.	Secure datanodes, user to run the datanode as
HADOOP_SSH_OPTS	HADOOP_SSH_OPTS	Extra ssh options.
HADOOP_LOG_DIR	HADOOP_LOG_DIR	Where log files are stored. $HADOOP_HOME/logs by default.
HADOOP_SECURE_DN_LOG_DIR	HADOOP_SECURE_DN_LOG_DIR	Where log files are stored in the secure data environment.
HADOOP_PID_DIR	HADOOP_PID_DIR	Directory where pid files are stored, /tmp by default.
HADOOP_SECURE_DN_PID_DIR	HADOOP_SECURE_DN_PID_DIR	Directory where pid files are stored, /tmp by default.
HADOOP_IDENT_STRING	HADOOP_IDENT_STRING	String representing this instance of hadoop. $USER by default.
MALLOC_ARENA_MAX	MALLOC_ARENA_MAX	Newer versions of glibc use an arena memory allocator that causes virtual memory usage to explode. This interacts badly with the many threads that we use in Hadoop. Tune the variable down to prevent vmem explosion.
Not in hadoop-env.sh.	HADOOP_MAPRED_LOG_DIR	
Not in hadoop-env.sh.	HADOOP_MAPRED_PID_DIR	
Not in hadoop-env.sh.	JAVA_LIBRARY_PATH	
Not in hadoop-env.sh.	JSVC_HOME	For starting the datanode on secure cluster.

17.4. Create Local Directories

You must create local directories for YARN on every note of NodeManagers (TaskTrackers) and set the appropriate permissions for your YARN log directories.

1. Set the permissions on the `yarn.nodemanager.local-dirs` directories. Run these commands on all DataNodes in your cluster.

```
chown -R yarn:hadoop ${yarn.nodemanager.local-dirs}
chmod 755 ${yarn.nodemanager.local-dirs}
```

   where `${yarn.nodemanager.local-dirs}` is your local directory.

2. Change permissions of directories of `yarn.nodemanager.log-dirs`. Run these commands on all DataNodes in your cluster.

```
chown -R yarn:hadoop ${yarn.nodemanager.log-dirs}
chmod 755 ${yarn.nodemanager.log-dirs}
```

   where `${yarn.nodemanager.log-dirs}` is your log directory.

3. Create directories for YARN_LOG_DIR and YARN_PID_DIR.

   a. Open `/etc/hadoop/conf/yarn-env.sh`

b. Write down your values for YARN_LOG_DIR and YARN_PID_DIR as the following instructions require values for the ${YARN_LOG_DIR} and ${YARN_PID_DIR}. For example in yarn-env.sh:

```
${YARN_LOG_DIR}=/grid/0/var/log/hadoop/yarn
 ${YARN_PID_DIR}=/grid/0/var/run/hadoop/yarn
```

4. Make directories for ${YARN_LOG_DIR} and ${YARN_PID_DIR} and set the appropriate permissions for them.

```
mkdir ${YARN_LOG_DIR}
chown yarn:hadoop ${YARN_LOG_DIR}
chown yarn:hadoop ${YARN_LOG_DIR}
mkdir ${YARN_PID_DIR}
chown yarn:hadoop ${YARN_PID_DIR}
chown yarn:hadoop ${YARN_PID_DIR}
```

17.5. Start HDFS

- Start HDFS.

  To start HDFS, run commands as the $HDFS_USER.

1. Start the NameNode. On the NameNode host machine, execute the following command:

```
su $HDFS_USER
                          export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop/sbin/hadoop-daemon.sh start namenode -upgrade
```

   On a large system, this can take a long time to complete.

Note

Run this command with the -upgrade option only once. After you have completed this step, you can bring up the NameNode using this command without including the -upgrade option.

2. Verify that the NameNode is up and running:

```
ps -ef|grep -i NameNode
```

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

```
su $HDFS_USER
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop/sbin/hadoop-daemon.sh start secondarynamenode
```

4. Verify that the Secondary NameNode is up and running:

```
ps -ef|grep  SecondaryNameNode
```

5. 

Note

If you are working on a non-secure DataNode, use $HDFS_USER. For a secure DataNode, use root.

Start DataNodes. On all the DataNodes, execute the following command:

```
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop/sbin/hadoop-daemon.sh start datanode
```

6. Verify that the DataNode process is up and running:

```
ps -ef|grep DataNode
```

7. Verify that Namenode can go out of safe mode.

```
hdfs dfsadmin -safemode wait
Safemode is OFF
```

In general, it takes 5-10 minutes to get out of safemode. For thousands of nodes with millions of data blocks, getting out of safemode could take up to 45 minutes.

17.5.1. Verify HDFS filesystem health

Analyze if the filesystem is healthy.

1. Run the fsck command on namenode as $HDFS_USER:

```
hadoop fsck / -files -blocks -locations > dfs-new-fsck-1.log
```

2. Run hdfs namespace and report.

   - List directories.

   ```
   hadoop dfs -lsr / > dfs-new-lsr-1.log
   ```

   - Run report command to create a list of DataNodes in the cluster.

   ```
   hadoop dfsadmin -report > dfs-new-report-1.log
   ```

3. Compare the namespace report before the upgrade and after the upgrade. Verify that user files exist after upgrade.

```
dfs-old-fsck-1.log < -- > dfs-new-fsck-1.log
dfs-old-lsr-1.log < -- > dfs-new-lsr-1.log
```

Note

You must do this comparison manually to catch all errors.

4. From the Namenode WebUI, see if all DataNodes are up and running.

```
http://<namenode>:50070
```

17.5.2. Finalize the Upgrade

You can start HDFS without finalizing the upgrade. When you are ready to discard your backup, you can finalize the upgrade.

Warning

You must verify your filesystem health before finalizing the upgrade. After you finalize an upgrade, you cannot roll back.

Run the following command as the $HDFS_USER:

```
hadoop dfsadmin -finalizeUpgrade
```

17.5.3. Create HDFS Directories

You must create the following HDFS directories after you upgrade:

- YARN NodeManager remote applications log
- HDFS Job History

To create the YARN NodeManager remote applications log

1. Open `/etc/hadoop/conf/yarn-site.xml`.
2. Write down the value of the `yarn.nodemanager.remote-app-log-dir` so that you can use it in place of the `${yarn.nodemanager.remote-app-log-dir}` variable in later examples. For example: `${yarn.nodemanager.remote-app-log-dir} = /app-logs`
3. Create the ${yarn.nodemanager.remote-app-log-dir} in HDFS.

   ```
   hdfs dfs -mkdir ${yarn.nodemanager.remote-app-log-dir}
   hdfs dfs -chown -R yarn:hadoop ${yarn.nodemanager.remote-app-log-dir}
   hdfs dfs -chmod -R 777 ${yarn.nodemanager.remote-app-log-dir}
   ```

4. Create a JobHistory directory in HDFS.
   a. Open `mapred-site.xml`.
   b. Write down the value of the `mapreduce.jobhistory.done-dir` so that you can use it in place of the `${mapreduce.jobhistory.done-dir}` variable in later examples.
   c. Write down the value of the `mapreduce.jobhistory.intermediate-done-dir` so that you can use it in place of the `${mapreduce.jobhistory.intermediate-done-dir}` variable in later examples.
   d. Create the JobHistory directories in HDFS.

      ```
      hadoop dfs -mkdir ${mapreduce.jobhistory.done-dir}
      hadoop dfs -mkdir ${mapreduce.jobhistory.intermediate-done-dir}
      hadoop dfs -chown -R mapred:hadoop ${mapreduce.jobhistory.done-dir}
      ```

```
hadoop dfs -chown -R mapred:hadoop ${mapreduce.jobhistory.intermediate-
done-dir}
hadoop dfs -chmod -R 777 ${mapreduce.jobhistory.done-dir}
hadoop dfs -chmod -R 777 ${mapreduce.jobhistory.intermediate-done-dir}
```

Note

You have to create the parent directories in HDFS by yourself. Grant the parent directories the same permissions.

17.5.4. Start YARN/MapReduce Services

To start YARN, run commands as a YARN user. To start MapReduce, run commands as a MapReduce user.

1. Start the ResourceManager on your previous JobTracker host.

```
su $YARN_USER
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh start resourcemanager
ps -ef | grep -i resourcemanager
```

2. Start the NodeManager on your previous TaskTracker hosts.

```
su $YARN_USER
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-yarn/sbin/yarn-daemon.sh start nodemanager
ps -ef | grep -i nodemanager
```

3. To start MapReduce, run the following commands as MapReduce user:

```
su $MAPREDUCE_USER
export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
/usr/lib/hadoop-mapreduce/sbin/mr-jobhistory-daemon.sh --config /etc/hadoop/
conf start historyserver
ps -ef | grep -i jobhistoryserver
```

17.5.5. Run Hadoop Smoke Tests

To smoke test your Hadoop upgrade, you can run the following MapReduce job.

Run this command as regular user. The job uses MapReduce to write 100MB of data into HDFS with RandomWriter. `hadoop jar /usr/lib/hadoop-mapreduce/*examples*.jar randomwriter -Dtest.randomwrite.total_bytes=100000000 test-after-upgrade` You should see messages similar to:

```
map 0% reduce 0%
…
map 100% reduce 100%
Job …. completed successfully
```

You just submitted your first MapReduce job in HDP 2.x. Good job! The next steps are to upgrade your other components.

Basic troubleshooting:

1. To find the number of active nodes and NodeManagers, access the ResourceManager web UI:

   ```
   http://<resource manager host>:8088/cluster/nodes
   ```

   The number of active nodes should be equal to the number of nodemanagers.

2. Error messages. Access the ApplicationMaster WebUI to view the container logs.

   a. Looking at your console logs for MapReduce job, there is a line in this format:

   ```
   13/10/02 17:57:21 INFO mapreduce.Job: The url to track the job: http://
   <resource manager host>:8088/proxy/application_1380673658357_0007/
   ```

   b. Go to the URL and select the job link.

   c. Select the logs link under ApplicationMaster table. It will redirect you to the container logs. Error messages display here.

17.6. Upgrade Apache ZooKeeper

1. Execute the following command on all the Apache ZooKeeper nodes:

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum upgrade zookeeper
     ```

   - For SLES:

     ```
     zypper update zookeeper
     ```

   - For Ubuntu:

     ```
     apt-get update zookeeper
     ```

2. Replace your configuration after upgrading. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.

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

   ```
   sudo su -l $ZOOKEEPER_USER -c "source /etc/zookeeper/conf/zookeeper-env.sh;
   export ZOOCFGDIR=/etc/zookeeper/conf; /usr/lib/zookeeper/bin/zkServer.sh
    start >> $ZOOKEEPER_LOG_DIR/zoo.out\"
   ```

   where

   - *$ZOOKEEPER_USER* is the ZooKeeper Service user. For example, `zookeeper`.
   - *$ZOOKEEPER_LOG_DIR* is the directory where ZooKeeper server logs are stored. For example, `/var/log/zookeeper`.

17.7. Upgrade Apache HBase

Depeding on your tolernace for downtime in your cluster, there are the following approaches for upgrading Apache HBase:

- Downtime Tolerance: you can tolerate more downtime to HBase for a detailed upgrade process and can shut down your HBase source cluster.
- Minimized Downtime: you must minimize downtime as much as possible during the upgrade process.

17.7.1. Downtime Tolerance

ipsum

1. Stop your HBase service
2. Shut down your source HBase cluster.

```
su hbase - -c "hbase-daemon.sh stop"
```

3. Shut down destination hbase cluster.

```
su hbase - -c "hbase-daemon.sh --config /etc/hbase/ stop regionserver"
```

4. Clean destination hbase cluster hdfs files and ZK data.
5. Distcp all hbase files from source hbase root directory to destination hbase cluster root folder

   Run upgrade step 4 and 5 from the above guide Upgrade should complete successfully

1. Upgrade HBase.

   Run the following commands on both the HBase Master and the RegionServers hosts.

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum upgrade hbase
     ```

     For SLES:

     ```
     zypper install hbase
     ```

   - For Ubuntu:

     ```
     apt-get install hbase
     ```

2. Replace your configuration after upgrading. Replace the HBase template configuration in `/etc/hbase/conf`.
3. Check for HFiles in V1 format. HBase 0.96.0 discontinues support for HFileV1. Before the actual upgrade, run the following command to check if there are HFiles in V1 format:

   ```
   hbase upgrade -check
   ```

   HFileV1 was a common format prior to HBase 0.94. You may see output similar to:

   ```
   Tables Processed:

   hdfs://localhost:41020/myHBase/.META.
   ```

```
hdfs://localhost:41020/myHBase/usertable
hdfs://localhost:41020/myHBase/TestTable
hdfs://localhost:41020/myHBase/t

Count of HFileV1: 2
HFileV1:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812/
family/249450144068442524
hdfs://localhost:41020/myHBase/usertable/ecdd3eaee2d2fcf8184ac025555bb2af/
family/249450144068442512

Count of corrupted files: 1
Corrupted Files:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812/
family/1
Count of Regions with HFileV1: 2
Regions to Major Compact:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812
hdfs://localhost:41020/myHBase/usertable/ecdd3eaee2d2fcf8184ac025555bb2af
```

When you run the upgrade check, if "Count of HFileV1" returns any files, start the HBase shell to use major compaction for regions that have HFileV1 format. For example in the sample output above, you must compact the `fa02dac1f38d03577bd0f7e666f12812` and `ecdd3eaee2d2fcf8184ac025555bb2af` regions.

4. As the HBase user, run an upgrade:

```
sudo su -l $HBASE_USER -c "hbase upgrade -execute"
```

You should see a completed Znode upgrade with no errors.

5. Start services. Run as root:

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

6. Check processes.

```
ps -ef | grep -i hmaster
ps -ef | grep -i hregion
```

17.7.2. Minimizing Downtime

ipsum

1. Shut down destination hbase cluster (assuming it's empty cluster)
2. Mark current time stamp Ts
3. Clean destination hbase cluster hdfs files and ZK data
4. Distcp all hbase files from source hbase root directory to destination hbase cluster root folder

5. Run upgrade step 4 and 5 from the above guide

6. Shut down source cluster.

```
su hbase - -c "hbase-daemon.sh --config /etc/hbase/ stop regionserver"
```

7. Use export to dump all data from source cluster since the above marked time stamp Ts.

```
$ bin/hbase org.apache.hadoop.hbase.mapreduce.Export <tablename> <outputdir>
 [<versions> [<starttime> [<endtime>]]]
```

8. Use import to get exported tail data into the destination cluster Route traffic to destination cluster

17.8. Upgrade Apache Hive and Apache HCatalog

1. Upgrade Apache Hive and Apache HCatalog. On the Hive and HCatalog host machines, execute the following commands:

   - For RHEL/CentOS:

     ```
     yum upgrade hive hcatalog
     ```

   - For SLES:

     ```
     zypper update hive hcatalog
     yast --update hcatalog hive
     ```

   - For Ubuntu:

     ```
     apt-get update hive hcatalog
     ```

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

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

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

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

   where

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

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

17.9. Upgrade Apache Oozie

1. Execute the following command on the Apache Oozie server and client machines:

   - For RHEL/CentOS:

```
yum install oozie
```

- For SLES:

```
zypper install oozie (if not already installed)
```

- For Ubuntu:

```
apt-get install oozie
```

2. You must replace your configuration after upgrading. Copy `/etc/oozie/conf` from the template to the conf directory on each oozie server and client.

3. Change the JDBC config to match your Oozie database. The entries to edit are:

```
oozie.service.JPAService.jdbc.driver
oozie.service.JPAService.jdbc.url
```

For example, for MySQL, use:

```
oozie.service.JPAService.jdbc.driver = com.mysql.jdbc.Driver
oozie.service.JPAService.jdbc.url = jdbc:mysql://$my_server:my_port/oozie?
createDatabaseIfNotExist=true
```

Note

If your are upgrading from HDP 2.0.6.0 to HDP 2.0.6.1, you do not need to re-install. Run the upgrade command instead:

- For RHEL/CentOS:

```
yum upgrade oozie
```

- For SLES:

```
zypper upgrade oozie
```

- For Ubuntu:

```
apt-get updrade oozie
```

4. Copy the JDBC jar to libext.

   a. Create the `/usr/lib/oozie/libext` directory.

```
cd /usr/lib/oozie
mkdir libext
```

   b. Grant read/write/execute access to all users for the libext directory.

```
chmod -R 777 /usr/lib/oozie/libext
```

5. Copy the JDBC jar of your Oozie database to the libext directory. For example, if you are using MySQL the `mysql-connector-java.jar` is found in `/usr/lib/oozie/libtool`.

6. Copy these files libext directory

```
cp /usr/lib/hadoop/lib/hadoop-lzo*.jar /usr/lib/oozie/libext
cp /usr/share/HDP-oozie/ext-2.2.zip /usr/lib/oozie/libext/
```

7. Extract share-lib.

```
cd /usr/lib/oozie
tar xzvf /usr/lib/oozie//oozie-sharelib.tar.gz
su -l hdfs -c "hdfs dfs -mkdir -p /user/oozie"
su -l hdfs -c "hdfs dfs -copyFromLocal /usr/lib/oozie/share /user/oozie/."
```

   You may see complaints that some files exist. This is an expected behavior.

```
su -l hdfs -c "hdfs dfs -chown oozie:hadoop /user/oozie"
su -l hdfs -c "hdfs dfs -chmod -R 755 /user/oozie"
```

8. Run upgrade as the Oozie user. Do not run as the root user to execute this.

```
su $OOZIE_USER
/usr/lib/oozie/bin/ooziedb.sh upgrade -run
```

9. Prepare the Oozie WAR file. Run as root:

```
sudo su -l oozie -c "/usr/lib/oozie/bin/oozie-setup.sh prepare-war -d /usr/
lib/oozie/libext"
```

   Look for console output to indicate success. For example, if you are using MySQL you should see something similar to:

```
INFO: Adding extension: libext/mysql-connector-java.jar
New Oozie WAR file with added 'JARs' at /var/lib/oozie/oozie-server/webapps/
oozie.war
```

10. Replace the content of `/user/oozie/share` in HDFS. On the Oozie server host:

    a. Extract the Oozie sharelib into a `tmp` folder.

```
mkdir -p /tmp/oozie_tmp
cp /usr/lib/oozie/oozie-sharelib.tar.gz /tmp/oozie_tmp
cd /tmp/oozie_tmp
tar xzvf oozie-sharelib.tar.gz
```

    b. Back up the`/user/oozie/share` folder in HDFS and then delete it. If you have any custom files in this folder back them up separately and then add them back after the share folder is updated.

```
su -l hdfs -c "hdfs dfs -copyToLocal /user/oozie/share /tmp/oozie_tmp/
oozie_share_backup"
su -l hdfs -c "hdfs dfs -rm -r /user/oozie/share"
```

    c. Add the latest share libs that you extracted in step 1. After you have added the files, modify ownership and ACL.

```
su -l hdfs -c "hdfs dfs -copyFromLocal /tmp/oozie_tmp/share /user/oozie/.
"
su -l hdfs -c "hdfs dfs -chown -R oozie:hadoop /user/oozie"
su -l hdfs -c "hdfs dfs -chmod -R 755 /user/oozie"
```

11. Set the `oozie.service.WorkflowAppService.system.libpath` in oozie-site.xml to the right path of sharelib in hdfs.

```
<property>
  <name>oozie.service.WorkflowAppService.system.libpath</name>
  <value>/user/$
  {user.name}
  /share/lib</value>
  <description>
  System library path to use for workflow applications.
  This path is added to workflow application if their job properties sets
 the property 'oozie.use.system.libpath' to true.
  </description>
</property>
```

12. Start Oozie. Run as root.

```
sudo su -l oozie -c "cd /grid/0/var/log/oozie; /usr/lib/oozie/bin/oozie-
start.sh"
```

13. Check processes.

```
ps -ef | grep -i oozie
```

17.10. Upgrade Apache WebHCat (Templeton)

1. Remove old Templeton packages. On the Templeton host machine, execute the following commands:

   - For RHEL/CentOS:

     ```
     yum remove templeton\*
     ```

   - For SLES:

     ```
     zypper remove templeton\*
     ```

   - For Ubuntu:

     ```
     apt-get remove templeton\*
     ```

2. Install WebHCat.

   - For RHEL/CentOS:

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

   - For SLES:

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

   - For Ubuntu:

     ```
     apt-get install webhcat-tar-hive webhcat-tar-pig
     ```

   Also see the instructions on manually deploying WebHCat instance provided here.

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

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

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

```
http://$WEBHCAT_HOST_MACHINE:50111/templeton/v1/status
```

5. Remove shared libraries from old Templeton installation. On the WebHCat host machine, execute the following command:

```
sudo su -l $HDFS_USER -c "hadoop dfs -rmr -skipTrash /apps/templeton"
rm -rf /usr/share/HDP-templeton
```

   where

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

6. You must replace your configuration after upgrading. Copy `/etc/webhcat/conf` from the template to the conf directory in webhcat hosts.

17.11. Upgrade Apache Pig

1. On all the Apache Pig clients, execute the following command:

   - For RHEL/CentOS:

```
yum upgrade pig
```

   - For SLES:

```
zypper update pig
```

   - For Ubuntu:

```
apt-get install pig
```

2. You must replace your configuration after upgrading. Copy `/etc/pig/conf` from the template to the conf directory in pig hosts.

17.12. Upgrade Apache Sqoop

Upgrade Apache Sqoop. On the Sqoop host machine, execute the following command:

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade sqoop
```

- For SLES:

```
zypper update sqoop
```

- For Ubuntu:

```
apt-get install sqoop
```

- You must replace your configuration after upgrading. Copy `/etc/sqoop/conf` from the template to the conf directory in sqoop hosts.

17.13. Upgrade Apache Flume

Upgrade Apache Flume. On the Flume host machine, execute the following command:

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade flume
```

- For SLES:

```
zypper update flume
zypper remove flume
zypper se -s flume
```

You should see Flume in the output. Install Flume:

```
zypper install flume
```

Important

When removing and installing packages, rename those files the `/conf` directory that have `.rpmsave` extension to original to retain the customized configs. Alternatively, you can also use the configuration files (under the `/conf` directory) you backed up before upgrading.

- You must replace your configuration after upgrading. Copy `/etc/flume/conf` from the template to the conf directory in Flume hosts.

- For Ubuntu:

```
apt-get install flume
```

17.13.1. Validate Flume

By default on installation Flume does not start running immediately. To validate, replace your default `conf/flume.conf` with the provdied `flume.conf`, restart Flume, and see if the data is flowing by examining the destination.

Use this `flume.conf` file:

```
1. Name the components on this agent
   a1.sources = r1
   a1.sinks = k1
   a1.channels = c1
2.Describe/configure the source
   a1.sources.r1.type = seq
3. Describe the sink
   a1.sinks.k1.type = file_roll
```

```
    a1.sinks.k1.channel = c1
    a1.sinks.k1.sink.directory = /tmp/flume
4. Use a channel which buffers events in memory
    a1.channels.c1.type = memory
5. Bind the source and sink to the channel
    a1.sources.r1.channels = c1
    a1.sinks.k1.channel = c1
```

After starting Flume, check `/tmp/flume` to see if there are any files there. The files should contain simple sequential numbers. After validating, stop Flume and revert changes to `flume.conf` to prevent your disk from filling up.

17.14. Upgrade Apache Mahout

Upgrade Apache Mahout. On the Mahout client machines, execute the following command:

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade mahout
```

- For SLES:

```
zypper remove mahout
zypper se -s mahout
```

You should see Mahout in the output. Install Mahout:

```
zypper install mahout
```

Important

When removing and installing packages, rename those files the `/conf` directory that have `.rpmsave` extension to original to retain the customized configs. Alternatively, you can also use the configuration files (under the `/conf` directory) you backed up before upgrading.

- For Ubuntu:

```
apt-get install mahout
```

- You must replace your configuration after upgrading. Copy `/etc/mahout/conf` from the template to the conf directory in mahout hosts.

17.14.1. Mahout Validation

To validate mahout:

1. Create a test user:

```
hadoop fs -put /tmp/sample-test.txt /user/testuser
```

2. Create a mahout test output directory:

```
hadoop fs -mkdir /user/testuser/mahouttest
```

3. Set up mahout to convert the plain text file `sample-test.txt` into a sequence file that is in the output directory `mahouttest`.

```
mahout seqdirectory --input /user/testuser/sample-test.txt --output /user/
testuser/mahouttest --charset utf-8
```

17.15. Upgrade Hue

For HDP 2, you must use Hue 2.3. Hue 2.2 supports HDP 1.x products.

Warning

If you are using the embedded SQLite database, you must perform a backup of the database before you upgrade Hue to prevent data loss. To make a backup copy of the database, simply do a "dump" and redirect the results to a file.

```
cd /var/lib/hue
sqlite3 desktop.db .dump > ~/desktop.bak
```

Execute the following command on all Hue Server host machines:

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade hue
```

- For SLES:

```
zypper update hue
```

Note

If you are using the embedded SQLite database, remember to restore your database after upgrade.

To restore the database from a backup, make sure the destination database is empty before copying. If necessary, rename or remove the current destination database. Then, copy your backup to the destination database. For example:

```
cd /var/lib/hue
mv desktop.db desktop.db.old
sqlite3 desktop.db < ~/desktop.bak
```

18. Setting Up Security for Manual Installs

This section provides information on enabling security for a manually installed version of HDP 2.

18.1. Preparing Kerberos

This section provides information on setting up Kerberos for an HDP 2 installation.

18.1.1. Kerberos Overview

To create secure communication among its various components, HDP 2 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.

18.1.2. Installing and Configuring the KDC

To use Kerberos with HDP 2 you can either use an existing KDC or install a new one just for use by HDP 2. The following gives a very high level description of the installation process. To get more information see RHEL documentation , CentOS documentation, SLES documentation. or Ubuntu documentation.

To install a new version of the server:

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

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

```
[On Ubuntu]
apt-get 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`.

[On SLES]

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

[On Ubuntu]

- `/etc/krb5.conf`
- `/var/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.

18.1.3. 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
```

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

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

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/CentOS/Oracle Linux]
   /sbin/service krb5kdc start
   /sbin/service kadmin start
   ```

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

   ```
   [On Ubuntu]
   /etc/init.d/krb5-kdc start
   /etc/init.d/kadmin start
   ```

18.1.4. Creating Service Principals and Keytab Files for HDP 2

Each service in HDP 2 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.

Step 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

The NameNode, Secondary NameNode, and Oozie require two principals each.

Note

If you are configuring High Availability (HA) for a Quorom-based NameNode, you must also generate a principle (jn/$FQDN) and keytab (jn.service.keytab) for each JournalNode. JournalNode also requires the keytab for its HTTP service. If the JournalNode is deployed on the same host as a NameNode, the same keytab file (spnego.service.keytab) can be used for both. In addition, HA requires two NameNodes. Both the active and standby NameNode require their own principle and keytab files. The service principles of the two NameNodes can share the same name, specified with the `dfs.namenode.kerberos.principal` and `dfs.secondary.namenode.kerberos.principal` properties in `hdfs-site.xml`, but the NameNodes still have different fully qualified domain names.

Table 18.1. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	nn/*$FQDN*
HDFS	NameNode HTTP	HTTP/*$FQDN*
HDFS	SecondaryNameNode	nn/*$FQDN*

Service	Component	Mandatory Principal Name
HDFS	SecondaryNameNode HTTP	HTTP/*$FQDN*
HDFS	DataNode	dn/*$FQDN*
MR2	History Server	jhs/*$FQDN*
MR2	History Server HTTP	HTTP/*$FQDN*
YARN	ResourceManager	rm/*$FQDN*
YARN	NodeManager	nm/*$FQDN*
Oozie	Oozie Server	oozie/*$FQDN*
Oozie	Oozie HTTP	HTTP/*$FQDN*
Hive	Hive Metastore HiveServer2	hive/*$FQDN*
Hive	WebHCat	HTTP/*$FQDN*
HBase	MasterServer	hbase/*$FQDN*
HBase	RegionServer	hbase/*$FQDN*
ZooKeeper	ZooKeeper	zookeeper/*$FQDN*
Nagios Server	Nagios	nagios/*$FQDN*
JournalNode Server[a]	JournalNode	jn/*$FQDN*

[a]Only required if you are setting up NameNode HA.

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

```
kadmin: addprinc -randkey dn/<datanode-host>@EXAMPLE.COM
```

Step 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 -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 18.2. Service Keytab File Names

Component	Principal Name	Mandatory Keytab File Name
NameNode	nn/*$FQDN*	nn.service.keytab
NameNode HTTP	HTTP/*$FQDN*	spnego.service.keytab
SecondaryNameNode	nn/*$FQDN*	nn.service.keytab
SecondaryNameNode HTTP	HTTP/*$FQDN*	spnego.service.keytab
DataNode	dn/*$FQDN*	dn.service.keytab
MR2 History Server	jhs/*$FQDN*	jhs.service.keytab
MR2 History Server HTTP	HTTP/*$FQDN*	spnego.service.keytab
YARN	rm/*$FQDN*	rm.service.keytab
YARN	nm/*$FQDN*	nm.service.keytab
Oozie Server	oozie/*$FQDN*	oozie.service.keytab
Oozie HTTP	HTTP/*$FQDN*	spnego.service.keytab
Hive Metastore HiveServer2	hive/*$FQDN*	hive.service.keytab
WebHCat	HTTP/*$FQDN*	spnego.service.keytab

Component	Principal Name	Mandatory Keytab File Name
HBase Master Server	`hbase/$FQDN`	`hbase.service.keytab`
HBase RegionServer	`hbase/$FQDN`	`hbase.service.keytab`
ZooKeeper	`zookeeper/$FQDN`	`zk.service.keytab`
Nagios Server	`nagios/$FQDN`	`nagios.service.keytab`
Journal Server[a]	`jn/$FQDN`	`jn.service.keytab`

[a]Only required if you are setting up NameNode HA.

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.

Step 3: 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.

18.2. Configuring HDP 2

This section provides information on configuring HDP 2 for Kerberos.

- Configuration Overview
- Creating Mappings Between Principals and UNIX Usernames
- Creating the Database and Setting Up the First Administrator
- Creating Principals and Keytab Files for HDP 2

18.2.1. Configuration Overview

Configuring HDP 2 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 2.

18.2.2. Creating Mappings Between Principals and UNIX Usernames

HDP 2 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.

18.2.2.1. 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.

18.2.2.1.1. 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`

18.2.2.1.2. 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`

18.2.2.1.3. 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`

18.2.2.2. 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
```

18.2.3. Adding Security Information to Configuration Files

To enable security on HDP 2, you must add optional information to various configuration files.

Before you begin, set JSVC_Home in `hadoop-env.sh`.

- For RHEL/CentOS/Oracle Linux:

```
export JSVC_HOME=/usr/libexec/bigtop-utils
```

- For SLES and Ubuntu:

```
export JSVC_HOME=/usr/lib/bigtop-utils
```

18.2.3.1. core-site.xml

To the `core-site.xml` file on *every* host in your cluster, you must add the following information:

Table 18.3. core-site.xml

Property Name	Property Value	Description
hadoop.security.authentication	kerberos	Set the authentication type for the cluster. Valid values are: simple or kerberos.
hadoop.rpc.protection	authentication; integrity; privacy	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 as well as authentication. privacy = authentication, integrity, and confidentiality; guarantees that data exchanged between client and server is encrypted and is not readable by a "man in the middle".
hadoop.security.authorization	true	Enable authorization for different protocols.
hadoop.security.auth_to_local	The mapping rules. For exampleRULE:[2:$1@$0] ([jt]t@.*EXAMPLE.COM)s/.*/ mapred/ RULE:[2:$1@$0] ([nd]n@.*EXAMPLE.COM)s/.*/ hdfs/ RULE:[2:$1@$0] (hm@.*EXAMPLE.COM)s/.*/ hbase/ RULE:[2:$1@$0] (rs@.*EXAMPLE.COM)s/.*/ hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. See Creating Mappings Between Principals and UNIX Usernames for more information.

The XML for these entries:

```
  <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.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/
        RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/hdfs/
```

```
        RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.*/hbase/
        RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/.*/hbase/
        DEFAULT</value> <description>The mapping from kerberos principal names
        to local OS user names.
</property>
```

18.2.3.2. hdfs-site.xml

To the `hdfs-site.xml` file on *every* host in your cluster, you must add the following information:

Table 18.4. hdfs-site.xml

Property Name	Property Value	Description
`dfs.permissions.enabled`	`true`	If `true`, permission checking in HDFS is enabled. If `false`, permission checking is turned off, but all other behavior is unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.
`dfs.permissions.supergroup`	`hdfs`	The name of the group of super-users.
`dfs.block.access.token.enable`	`true`	If `true`, access tokens are used as capabilities for accessing datanodes. If `false`, no access tokens are checked on accessing datanodes.
`dfs.namenode.kerberos.principal`	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the NameNode.
`dfs.secondary.namenode.kerberos.principal`	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the secondary NameNode.
`dfs.web.authentication.kerberos.principal`	HTTP/_HOST@EXAMPLE.COM	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.
`dfs.web.authentication.kerberos.keytab`	`/etc/security/keytabs/spnego.service.keytab`	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
`dfs.datanode.kerberos.principal`	dn/_HOST@EXAMPLE.COM	The Kerberos principal that the DataNode runs as. "_HOST" is replaced by the real host name .
`dfs.namenode.keytab.file`	`/etc/security/keytabs/nn.service.keytab`	Combined keytab file containing the NameNode service and host principals.
`dfs.secondary.namenode.keytab.file`	`/etc/security/keytabs/nn.service.keytab`	Combined keytab file containing the NameNode service and host principals. <question?>
`dfs.datanode.keytab.file`	`/etc/security/keytabs/dn.service.keytab`	The filename of the keytab file for the DataNode.
`dfs.https.port`	50470	The https port to which the NameNode binds
`dfs.namenode.https-address`	Example: ip-10-111-59-170.ec2.internal:50470	The https address to which the NameNode binds
`dfs.datanode.data.dir.perm`	750	The permissions that must be set on the `dfs.data.dir` directories. The DataNode will not come up if all existing `dfs.data.dir` directories do

Property Name	Property Value	Description
		not have this setting. If the directories do not exist, they will be created with this permission
dfs.cluster.administrators	hdfs	ACL for who all can view the default servlets in the HDFS
dfs.namenode.kerberos.internal.spnego.principal	${dfs.web.authentication.kerberos.principal}	
dfs.secondary.namenode.kerberos.internal.spnego.principal	${dfs.web.authentication.kerberos.principal}	

The XML for these entries:

```
<property>
        <name>dfs.permissions</name>
        <value>true</value>
        <description> If "true", enable permission checking in
        HDFS. If "false", permission checking is turned
        off, but all other behavior is
        unchanged. Switching from one parameter value to the other does
        not change the mode, owner or group of files or
        directories. </description>
</property>

<property>
        <name>dfs.permissions.supergroup</name>
        <value>hdfs</value>
        <description>The name of the group of
        super-users.</description>
</property>

<property>
        <name>dfs.namenode.handler.count</name>
        <value>100</value>
        <description>Added to grow Queue size so that more
        client connections are allowed</description>
</property>

<property>
        <name>ipc.server.max.response.size</name>
        <value>5242880</value>
</property>

<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>ip-10-72-235-178.ec2.internal:50090</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>ip-10-111-59-170.ec2.internal:50470</value>
        <description>The https address where namenode binds</description>
</property>

<property>
        <name>dfs.datanode.data.dir.perm</name>
        <value>750</value>
        <description>The permissions that should be there on
        dfs.data.dir directories. The datanode will not come up if the
        permissions are different on existing dfs.data.dir directories. If
        the directories don't exist, they will be created with this
        permission.</description>
</property>

<property>
        <name>dfs.access.time.precision</name>
        <value>0</value>
        <description>The access time for HDFS file is precise upto this
        value.The default value is 1 hour. Setting a value of 0
        disables access times for HDFS.
        </description>
</property>

<property>
        <name>dfs.cluster.administrators</name>
        <value> hdfs</value>
        <description>ACL for who all can view the default
        servlets in the HDFS</description>
</property>

<property>
        <name>ipc.server.read.threadpool.size</name>
        <value>5</value>
        <description></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>
```

In addition, you must set the user on all secure DataNodes:

```
export HADOOP_SECURE_DN_USER=hdfs
export HADOOP_SECURE_DN_PID_DIR=/grid/0/var/run/hadoop/$HADOOP_SECURE_DN_USER
```

18.2.3.3. mapred-site.xml

To the `mapred-site.xml` file on *every* host in your cluster, you must add the following information:

Table 18.5. mapred-site.xml

Property Name	Property Value	Description	Final
mapreduce.jobtracker.kerberos.principal	jt/_HOST@EXAMPLE.COM	Kerberos principal name for the JobTracker	
mapreduce.tasktracker.kerberos.principal	tt/_HOST@EXAMPLE.COM	Kerberos principal name for the TaskTracker. _HOST" is replaced by the host name of the task tracker.	
hadoop.job.history.user.location	none		true
mapreduce.jobtracker.keytab.file	/etc/security/keytabs/jt.service.keytab	The keytab for the JobTracker principal	
mapreduce.tasktracker.keytab.file	/etc/security/keytabs/tt.service.keytab	The keytab for the Tasktracker principal	
mapreduce.jobtracker.staging.root.dir	/user	The path prefix for the location of the the staging directories. The next level is always the user's name. It is a path in the default file system	
mapreduce.tasktracker.group	hadoop	The group that the task controller uses for accessing the task controller. The mapred user must be a member and users should **not** be members. <question?>	
mapreduce.jobtracker.split.metainfo.maxsize	50000000	If the size of the split metainfo file is larger than this value, the JobTracker will fail the job during initialization.	true
mapreduce.history.server.embedded	false	Should the Job History server be embedded within the JobTracker process	true
mapreduce.history.server.http.address **Note**: cluster variant	Example: ip-10-111-59-170.ec2.internal:51111		

Property Name	Property Value	Description	Final
mapreduce.jobhistory.kerberos.principal **Note**: cluster variant	jt/_HOST@EXAMPLE.COM	Kerberos principal name for JobHistory. This must map to the same user as the JT user.	true
mapreduce.jobhistory.keytab.file **Note**: cluster variant	/etc/security/keytabs/jt.service.keytab	The keytab for the JobHistory principal	
mapred.jobtracker.blacklist.fault-timeout-window	Example default 180	3-hour sliding window - the value is specified in minutes.	
mapred.jobtracker.blacklist.fault-bucket-width	Example default 15	15-minute bucket size - the value is specified in minutes.	
mapred.queue.names	default	Comma separated list of queues configured for this jobtracker.	

The XML for these entries:

```
<property>
        <name>mapreduce.jobtracker.kerberos.principal</name>
        <value>jt/_HOST@EXAMPLE.COM</value>
        <description> JT
        user name key.  </description>
</property>

<property>
        <name>mapreduce.tasktracker.kerberos.principal</name>
        <value>tt/_HOST@EXAMPLE.COM</value>
        <description>tt
        user name key. "_HOST" is replaced by the host name of the task
 tracker.
        </description>
</property>

<property>
        <name>hadoop.job.history.user.location</name>
        <value>none</value>
        <final>true</final>
</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 task
        tracker</description>
</property>

<property>
        <name>mapreduce.jobtracker.staging.root.dir</name>
        <value>/user</value>
```

```
        <description>The Path prefix for where the staging
        directories should be placed. The next level is always the user's
 name. It
        is a path in the default file system.</description>
</property>

<property>
        <name>mapreduce.tasktracker.group</name>
        <value>hadoop</value>
        <description>The group that the task controller uses for accessing the
 task controller.
        The mapred user must be a member and users should *not* be
        members.</description>
</property>

<property>
        <name>mapreduce.jobtracker.split.metainfo.maxsize</name>
        <value>50000000</value>
        <final>true</final>
        <description>If the size of the split metainfo file is larger than
 this, the JobTracker
        will fail the job during
        initialize.
        </description>
</property>

<property>
        <name>mapreduce.history.server.embedded</name>
        <value>false</value>
        <description>Should job history server be embedded within Job tracker
 process</description>
        <final>true</final>
</property>

<property>
        <name>mapreduce.history.server.http.address</name>
        <!--cluster variant -->
        <value>ip-10-111-59-170.ec2.internal:51111</value>
        <description>Http address of the history server</description>
        <final>true</final>
</property>

<property>
        <name>mapreduce.jobhistory.kerberos.principal</name>
        <!--cluster variant -->
        <value>jt/_HOST@EXAMPLE.COM</value>
        <description>Job history user name key. (must map to same user as JT
 user)</description>
</property>

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

<property>
        <name>mapred.jobtracker.blacklist.fault-timeout-window</name>
```

```
        <value>180</value>
        <description>     3-hour
        sliding window (value is in minutes)
        </description>
</property>

<property>
        <name>mapred.jobtracker.blacklist.fault-bucket-width</name>
        <value>15</value>
        <description>
        15-minute bucket size (value is in minutes)
        </description>
</property>

<property>
        <name>mapred.queue.names</name>
        <value>default</value>   <description>
        Comma separated list of queues configured for this jobtracker.</
description>
</property>
```

18.2.3.4. hbase-site.xml

For Hbase to run on a secured cluster, Hbase must be able to authenticate itself to HDFS. To the `hbase-site.xml` file on your HBase server, you must add the following information. There are no default values; the following are all only examples:

Table 18.6. hbase-site.xml

Property Name	Property Value	Description
`hbase.master.keytab.file`	`/etc/security/keytabs/hm.service.keytab`	The keytab for the HMaster service principal
`hbase.master.kerberos.principal`	hm/_HOST@EXAMPLE.COM	The Kerberos principal name that should be used to run the HMaster process. If _H0ST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
`hbase.regionserver.keytab.file`	`/etc/security/keytabs/rs.service.keytab`	The keytab for the HRegionServer service principal
`hbase.regionserver.kerberos.principal`	rs/_H0ST@EXAMPLE.COM	The Kerberos principal name that should be used to run the HRegionServer process. If _H0ST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
`hbase.superuser`	hbase	Comma-separated List of users or groups that are allowed full privileges, regardless of stored ACLs, across the cluster. Only used when HBase security is enabled.
`hbase.coprocessor.region.classes`		Comma-separated list of Coprocessors that are loaded by default on all tables. For any override coprocessor method, these classes will be called in order. After implementing your own Coprocessor, just put it in HBase's classpath and add the fully qualified

Property Name	Property Value	Description
		class name here. A coprocessor can also be loaded on demand by setting HTableDescriptor.
hbase.coprocessor.master.classes		Comma-separated list of org.apache.hadoop.hbase.coprocessor.MasterObserver coprocessors that are loaded by default on the active HMaster process. For any implemented coprocessor methods, the listed classes will be called in order. After implementing your own MasterObserver, just put it in HBase's classpath and add the fully qualified class name here.

The XML for these entries:

```
  <property>
        <name>hbase.master.keytab.file</name>
        <value>/etc/security/keytabs/hm.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>hm/_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/rs.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>rs/_HOST@EXAMPLE.COM</value>
        <description>Ex. "hbase/_HOST@EXAMPLE.COM".
        The kerberos principal name that
        should be used to run the HRegionServer 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.  An entry for this principal must exist
        in 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></value>
        <description>A comma-separated list of Coprocessors that are loaded
        by default on all tables. For any override coprocessor method, these
 classes
        will be called in order. After implementing your own Coprocessor,
        just put it in HBase's classpath and add the fully qualified class
 name here. A
        coprocessor can also be loaded on demand by setting HTableDescriptor.

        </description>
</property>

<property>
        <name>hbase.coprocessor.master.classes</name>
        <value></value>
        <description>A comma-separated list of
        org.apache.hadoop.hbase.coprocessor.MasterObserver coprocessors that
        are loaded by default on the active HMaster process. For any
 implemented
        coprocessor methods, the listed classes will be called in order.
        After implementing your own MasterObserver, just put it in HBase's
        classpath and add the fully qualified class name here.
        </description>
</property>
```

18.2.3.5. hive-site.xml

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

Table 18.7. hive-site.xml

Property Name	Property Value	Description
`hive.metastore.sasl.enabled`	`true`	If `true`, the Metastore Thrift interface will be secured with SASL and clients must authenticate with Kerberos
`hive.metastore.kerberos.keytab.file`	`/etc/security/keytabs/hive.service.keytab`	The keytab for the Metastore Thrift service principal
`hive.metastore.kerberos.principal`	`hive/_HOST@EXAMPLE.COM`	The service principal for the Metastore Thrift server. If _H0ST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.

Property Name	Property Value	Description
hive.metastore.cache.pinobjtypes	Table,Database,Type,FieldSchema,Order	Comma-separated Metastore object types that should be pinned in the cache

The XML for these entries:

```
<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.metastore.cache.pinobjtypes</name>
        <value>Table,Database,Type,FieldSchema,Order</value>
        <description>List of comma separated metastore object types that
 should be pinned in
        the cache</description>
</property>
```

18.2.3.5.1. oozie-site.xml

To the `oozie-site.xml` file, you must add the following information:

Table 18.8. oozie-site.xml

Property Name	Property Value	Description
oozie.service.AuthorizationService.security.enabled	true	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.
oozie.service.HadoopAccessorService.kerberos.enabled	true	Indicates if Oozie is configured to use Kerberos
local.realm	EXAMPLE.COM	Kerberos Realm used by Oozie and Hadoop. Using 'local.realm' to be aligned with Hadoop configuration.
oozie.service.HadoopAccessorService.keytab.file	/etc/security/keytabs/ oozie.service.keytab	The keytab for the Oozie service principal.

Property Name	Property Value	Description
oozie.service.HadoopAccessorService.kerberos.principal	oozie/_HOST@EXAMPLE.COM	Kerberos principal for Oozie service
oozie.authentication.type	kerberos	
oozie.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	Whitelisted job tracker for Oozie service
oozie.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	Location of the Oozie user keytab file.
oozie.service.HadoopAccessorService.nameNode.whitelist		
oozie.authentication.kerberos.name.rules	RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/mapred/ RULE:[2:$1@$0]([nd]n@.*EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](hm@.*EXAMPLE.COM)s/.*/hbase/ RULE:[2:$1@$0](rs@.*EXAMPLE.COM)s/.*/hbase/ DEFAULT	The mapping from Kerberos principal names to local OS user names. See Creating Mappings Between Principals and UNIX Usernames for more information.

18.2.3.5.2. webhcat-site.xml

To the `webhcat-site.xml` file, you must add the following information:

Table 18.9. webhcat-site.xml

Property Name	Property Value	Description
templeton.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	
templeton.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	
templeton.kerberos.secret	secret	

18.3. 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 for 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)

18.3.1. Configure HBase Master

Edit *$HBASE_CONF_DIR*/`hbase-site.xml` file on your HBase Master server to add the following information (*$HBASE_CONF_DIR* is the directory to store the HBase configuration files. For example, `/etc/hbase/conf`) :

Note

There are no default values. The following are all 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.

18.3.2. 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`.

18.3.3. 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
```

18.3.4. Configure secure client side access for 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_CONF_DIR/* `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>
```

18.3.5. Optional: Configure client-side operation for secure operation - Thrift Gateway

Add the following to the *$HBASE_CONF_DIR*`/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.

18.3.6. Optional: Configure client-side operation for secure operation - REST Gateway

Add the following to the *$HBASE_CONF_DIR*`/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.

18.3.7. 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. See 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').

19. Uninstalling HDP

Use the following instructions to uninstall HDP:

1. Stop all the installed HDP services.
2. If HCatalog is installed, execute the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove hcatalog\*
     ```

   - SLES:

     ```
     zypper remove hcatalog\*
     ```

   - Ubuntu:

     ```
     sudo apt-get remove hcatalog\*
     ```

3. If Hive is installed, execute the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove hive\*
     ```

   - SLES:

     ```
     zypper remove hive\*
     ```

   - Ubuntu:

     ```
     sudo apt-get remove hive\*
     ```

4. If HBase is installed, execute the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove hbase\*
     ```

   - SLES:

     ```
     zypper remove hbase\*
     ```

   - Ubuntu:

     ```
     sudo apt-get remove hbase\*
     ```

5. If ZooKeeper is installed, execute the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove zookeeper\*
     ```

   - SLES:

     ```
     zypper remove zookeeper\*
     ```

- Ubuntu:

```
sudo apt-get remove zookeeper\*
```

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

   - For RHEL/CentOS/Oracle Linux:

   ```
   yum remove oozie\*
   ```

   - SLES:

   ```
   zypper remove oozie\*
   ```

   - Ubuntu:

   ```
   sudo apt-get remove oozie\*
   ```

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

   - For RHEL/CentOS/Oracle Linux:

   ```
   yum remove pig\*
   ```

   - SLES:

   ```
   zypper remove pig\*
   ```

   - Ubuntu:

   ```
   sudo apt-get remove pig\*
   ```

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

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

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

   ```
   yum remove hadoop\*
   ```

10. 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\*
    ```