Hortonworks Data Platform

Installing HDP Manually

(May 2, 2014)

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. Meet minimum system requirements
2. Configure the remote repositories
3. Decide on deployment type
4. Collect information
5. Prepare the environment
6. Download companion files
7. Define environment parameters
8. Optional - Create system users and groups
9. Determine HDP Memory Configuration Settings

1.1. Meet Minimum System Requirements

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

- Hardware Recommendations
- Operating System Requirements
- Software Requirements
- Metastore Database Requirements
- JDK Recommendations

1.1.1. 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.

1.1.2. 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

1.1.3. Software Requirements

On each of your hosts:

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

1.1.4. Metastore Database Requirements

If you are installing Hive and HCatalog or installing Oozie, you must install a database to store metadata information in the metastore. You can either use an existing database instance or install a new instance manually. HDP supports the following databases for the metastore:

- Postgres 8.x, 9.x
- MySQL 5.x
- Oracle 11g r2
- SQL Server 2012, 2014

The database administrator must create the following databases users for Hive and/or Oozie:

- 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

By default, Hive uses the Derby database for the metastore. However, Derby is not supported for production systems.

- Installing and Configuring Postgres 8.x and 9.x
- Installing and Configuring MySQL 5.x
- Installing and Configuring Oracle 11g r2

1.1.4.1. Installing and Configuring PostgresSQL

The following instructions explain how to install PostgresSQL as the metastore database. See your third-party documentation for instructions on how to install other supported databases.

To install a new instance of PostgresSQL:

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. On the Hive Metastore host, install the connector.

   a. Install the connector.

   RHEL/CentOS/Oracle Linux

   ```
   yum install postgresql-jdbc*
   ```

   SLES

   ```
   zypper install -y postgresql-jdbc
   ```

   b. Copy the connector .jar file to the Java share directory.

   ```
   cp /usr/share/pgsql/postgresql-*.jdbc3.jar /usr/share/java/postgresql-
   jdbc.jar
   ```

   c. Confirm that .jar is in the Java share directory.

   ```
   ls /usr/share/java/postgresql-jdbc.jar
   ```

   d. Change the access mode of the .jar file to 644.

   ```
   chmod 644 /usr/share/java/postgresql-jdbc.jar
   ```

6. Load the Hive schema:

```
psql -U $HIVEUSER -d $HIVEDATABASE
\connect $HIVEDATABASE;
\i hive-schema-0.13.0.postgres.sql;
```

Note

The Hive schema is located at `/usr/lib/hive/scripts/metastore/upgrade/postgres`.

1.1.4.2. Installing and Configure MySQL

The following instructions explain how to install MySQL as the metastore database. See your third-party documentation for instructions on how to install other supported databases.

To install a new instance of MySQL:

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/Oracle Linux:

```
yum install mysql-server
```

For SLES:

```
zypper install mysql-server
```

For Ubuntu:

```
apt-get install mysql-server
```

3. Start the instance.

   For RHEL/CentOS/Oracle Linux:

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

   For SLES:

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

   For Ubuntu:

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

4. Set the `root` user password using the following command format:

   ```
   mysqladmin -u root password $mysqlpassword
   ```

   For example, to set the password to "root":

   ```
   mysqladmin -u root password root
   ```

5. Remove unnecessary information from log and STDOUT.

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

6. Now that the root password has been set, you can use the following command to log in to MySQL as `root`:

   ```
   mysql -u root -proot
   ```

   As `root`, create the “dbuser” and grant it adequate privileges. This user provides access to the Hive metastore. Use the following series of commands (shown here with the returned responses) to create "dbuser" with password "dbuser".

   ```
   [root@c6402 /]# mysql -u root -proot
   Welcome to the MySQL monitor.  Commands end with ; or \g.
   Your MySQL connection id is 11
   Server version: 5.1.73 Source distribution

   Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

   Oracle is a registered trademark of Oracle Corporation and/or its
   affiliates. Other names may be trademarks of their respective
   owners.

   Type 'help;' or '\h' for help. Type '\c' to clear the current input
    statement.
   ```

```
mysql> CREATE USER 'dbuser'@'localhost' IDENTIFIED BY 'dbuser';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'localhost';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE USER 'dbuser'@'%' IDENTIFIED BY 'dbuser';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'localhost' WITH GRANT
 OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON *.* TO 'dbuser'@'%' WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

7. Use the `exit` command to exit MySQL.
8. You should now be able to reconnect to the database as "dbuser" using the following command:

   ```
   mysql -u dbuser -pdbuser
   ```

   After testing the "dbuser" login, use the `exit` command to exit MySQL.
9. 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*
     ```

10. Load the Hive database schema.

    ```
    mysql $HIVEUSER/$HIVEPASSWORD < hive-schema-0.13.0.mysql.sql
    ```

1.1.4.3. Installing and Configuring Oracle

To set up Oracle for use with Hive:

1. On the Hive Metastore host, install the appropriate JDBC .jar file.
   a. Download the Oracle JDBC (OJDBC) driver from http://www.oracle.com/technetwork/database/features/jdbc/index-091264.html.

b. Select `Oracle Database 11g Release 2 - ojdbc6.jar`.

c. Copy the .jar file to the Java share directory.

```
cp ojdbc6.jar /usr/share/java
```

d. Make sure the .jar file has the appropriate permissions - 644.

2. Create a user for Hive and grant it permissions.

- Using the Oracle database admin utility:

```
# sqlplus sys/root as sysdba
CREATE USER $HIVEUSER IDENTIFIED BY $HIVEPASSWORD;
GRANT SELECT_CATALOG_ROLE TO $HIVEUSER;
GRANT CONNECT, RESOURCE TO $HIVEUSER;
QUIT;
```

- Where $HIVEUSER is the Hive user name and $HIVEPASSWORD is the Hive user password.

3. Load the Hive database schema.

```
sqlplus $HIVEUSER/$HIVEPASSWORD < hive-schema-0.13.0.oracle.sql
```

Note

The hive schema is located at `/usr/lib/hive/scripts/metastore/upgrade/oracle/`.

1.1.5. JDK Requirements

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

- Oracle JDK 1.7 64-bit update 51 or higher
- Oracle JDK 1.6 update 31 64-bit

Note

Deprecated as of HDP 2.1

- OpenJDK 7 64-bit

1.1.5.1. Oracle JDK 7 update 51

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. Navigate to the `usr/java` folder. If this folder does not already exist, create the folder:

```
mkdir usr/java
cd usr/java
```

5. Download the Oracle 64-bit JDK (jdk-7u51-linux-x64.tar.gz) from the Oracle download site. Open a web browser and navigate to `http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase7-521261.html`.

   Accept the license agreement and download the file labeled "jdk-7u51-linux-x64.tar.gz".

   Note

   The label on the download page is "jdk-7u51-linux-x64.tar.gz", but the actual name of the file is "jdk-7u51-linux-x64.gz".

6. Copy the downloaded `jdk-7u51-linux-x64.gz` file to the `/usr/java` folder.

7. Navigate to the `/usr/java` folder and extract the `jdk-7u51-linux-x64.gz` file.

```
cd /usr/java
tar zxvf jdk-7u51-linux-x64.gz
```

   The JDK files will be extracted into a `usr/java/jdk1.7.0_51` directory.

8. Create a symbolic link (symlink) to the JDK.

```
ln -s /usr/java/jdk1.7.0_51 /usr/java/default
```

9. Set the JAVA_H0ME and PATH environment variables.

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

10. Verify that Java is installed in your environment by running the following command:

```
java -version
```

    You should see the following output:

```
java version "1.7.0_51"
Java(TM) SE Runtime Environment (build 1.7.0_51-b18)
Java HotSpot(TM) 64-Bit Server VM (build 24.51-b03, mixed mode)
```

1.1.5.2. Oracle JDK 1.6 update 31 (Deprecated)

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. Open a web browser and navigate to `http://www.oracle.com/technetwork/java/javase/downloads/java-archive-downloads-javase6-419409.html`.

   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_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 "1.6.0_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)
```

1.1.5.3. OpenJDK 7

Note

OpenJDK7 on HDP 2.1 does not work if you are using SLES as your OS.

Use the following instructions to manually install OpenJDK 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 OpenJDK 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/lib/jvm/java-1.7.0-openjdk-1.7.0.51.x86_64 /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)
```

1.1.6. 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.

1.2. 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 file `hdp.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/GA/2.1-latest/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - For RHEL/CentOS 6:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.1-latest/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - For SLES:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/suse11/2.x/GA/2.1-latest/hdp.repo -O /etc/zypp/repos.d/hdp.repo
     ```

2. Confirm the HDP repository is configured.

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum repolist
     ```

     You should see something like this. Verify that you have HDP-2.1.0.0 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-2.1.0.0 Hortonworks Data Platform Version - HDP-2.1.0.0 enabled: 53
     ```

   - For SLES:

     ```
     zypper repos
     ```

1.3. 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.

1.4. 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 must be granted ALL PRIVILEGES on that instance.

1.5. Prepare the Environment

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

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

1.5.1. Enable NTP on the Cluster

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

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
   ```

1.5.2. Check DNS

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

Note

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

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), set up a 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.

1.5.3. 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.

1.5.4. Disable IPTables

For Ambari to communicate during setup with the hosts it deploys to and manages, certain ports must be open and available. The easiest way to do this is to temporarily disable `iptables`.

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

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

If the security protocols at your installation do not allow you to disable `iptables`, you can proceed with them on, as long as all of the relevant ports are open and available. If you plan to run with them enabled, see Configuring Ports (for the 2.x stack) for more information on the necessary ports per component.

During the Ambari Server setup process, Ambari checks to see if `iptables` is running. If it is, a warning prints to remind you to check that the necessary ports are open and available. The **Host Confirm** step of the Cluster Install Wizard will also issue a warning for each host that has has `iptables` running.

Important

If you leave `iptables` enabled **and** do not set up the necessary ports, the cluster installation will fail.

1.6. Download Companion Files

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

```
wget http://public-repo-1.hortonworks.com/HDP/tools/2.1.2.1/
hdp_manual_install_rpm_helper_files-2.1.2.471.tar.gz
```

Hortonworks strongly recommends that you edit and source the files included in the companion files.

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;
```

1.7. Define Environment Parameters

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

1. Define directories.

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

   Note

   The `scripts.zip` file you downloaded in Download Companion Files includes a script, `directories.sh`, for setting directory environment parameters.

We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.1. Define Directories for Core Hadoop

Hadoop Service	Parameter	Definition

HDFS	`DFS_NAME_DIR`	Space separated list of directories where NameNode should store the file system image.
		
		For example,
		
		`/grid/hadoop/hdfs/nn`
		
		`/grid1/hadoop/hdfs/nn`

HDFS	`DFS_DATA_DIR`	Space separated list of directories where DataNodes should store the blocks.
		
		For example,
		
		`/grid/hadoop/hdfs/dn`
		
		`/grid1/hadoop/hdfs/dn`
		
		`/grid2/hadoop/hdfs/dn`

HDFS	`FS_CHECKPOINT_DIR`	Space separated list of directories where SecondaryNameNode should store the checkpoint image.
		
		For example,
		
		`/grid/hadoop/hdfs/snn`
		
		`/grid1/hadoop/hdfs/snn`
		
		`/grid2/hadoop/hdfs/snn`

HDFS	`HDFS_LOG_DIR`	Directory for storing the HDFS logs. This directory name is a combination of a directory and the *$HDFS_USER*. For example, `/var/log/hadoop/hdfs` where `hdfs` is the *$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`

Hadoop Service	Parameter	Definition
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_LOCAL_LOG_DIR	Space-separated list of directories where YARN will store container log data. /grid/hadoop/yarn/logs /grid1/hadoop/yarn/log
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*.
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 1.2. 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.

Hadoop Service	Parameter	Definition
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.
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, /etc/sqoop/conf.

If you use the Companion files, the following provides a snapshot of how your directories.sh file should look after you edit the TODO variables:

```
#!/bin/sh

#
```

```
# Directories Script
#
# 1. To use this script, you must edit the TODO variables below for your
 environment.
#
# 2. Warning: Leave the other parameters as the default values. Changing
 these default values will require you to
# change values in other configuration files.
#

#
# Hadoop Service - HDFS
#

# Space separated list of directories where NameNode will store file system
 image. For example, /grid/hadoop/hdfs/nn /grid1/hadoop/hdfs/nn
DFS_NAME_DIR="/grid/0/hadoop/hdfs/nn";

# Space separated list of directories where DataNodes will store the blocks.
 For example, /grid/hadoop/hdfs/dn /grid1/hadoop/hdfs/dn /grid2/hadoop/hdfs/
dn
DFS_DATA_DIR="/grid/0/hadoop/hdfs/dn";

# Space separated list of directories where SecondaryNameNode will store
 checkpoint image. For example, /grid/hadoop/hdfs/snn /grid1/hadoop/hdfs/
snn /grid2/hadoop/hdfs/snn
FS_CHECKPOINT_DIR="/grid/0/hadoop/hdfs/snn";



# Directory to store the HDFS logs.
HDFS_LOG_DIR="/var/log/hadoop/hdfs";

# Directory to store the HDFS process ID.
HDFS_PID_DIR="/var/run/hadoop/hdfs";

# Directory to store the Hadoop configuration files.
HADOOP_CONF_DIR="/etc/hadoop/conf";

#
# Hadoop Service - YARN
#

# Space-separated list of directories where YARN will store temporary data.
 For example, /grid/hadoop/yarn/local /grid1/hadoop/yarn/local /grid2/
hadoop/yarn/local
YARN_LOCAL_DIR="/grid/0/hadoop/yarn/local";

# Directory to store the YARN logs.
YARN_LOG_DIR="/var/log/hadoop/yarn";

# Space-separated list of directories where YARN will store container log
 data. For example, /grid/hadoop/yarn/logs /grid1/hadoop/yarn/logs /grid2/
hadoop/yarn/logs
YARN_LOCAL_LOG_DIR="/grid/0/hadoop/yarn/logs";

# Directory to store the YARN process ID.
YARN_PID_DIR="/var/run/hadoop/yarn";

#
```

```
# Hadoop Service - MAPREDUCE
#

# Directory to store the MapReduce daemon logs.
MAPRED_LOG_DIR="/var/log/hadoop/mapreduce";

# Directory to store the mapreduce jobhistory process ID.
MAPRED_PID_DIR="/var/run/hadoop/mapreduce";

#
# Hadoop Service - Hive
#

# Directory to store the Hive configuration files.
HIVE_CONF_DIR="/etc/hive/conf";

# Directory to store the Hive logs.
HIVE_LOG_DIR="/var/log/hive";

# Directory to store the Hive process ID.
HIVE_PID_DIR="/var/run/hive";

#
# Hadoop Service - WebHCat (Templeton)
#

# Directory to store the WebHCat (Templeton) configuration files.
WEBHCAT_CONF_DIR="/etc/hcatalog/conf/webhcat";

# Directory to store the WebHCat (Templeton) logs.
WEBHCAT_LOG_DIR="/var/log/webhcat";

# Directory to store the WebHCat (Templeton) process ID.
WEBHCAT_PID_DIR="/var/run/webhcat";

#
# Hadoop Service - HBase
#

# Directory to store the HBase configuration files.
HBASE_CONF_DIR="/etc/hbase/conf";

# Directory to store the HBase logs.
HBASE_LOG_DIR="/var/log/hbase";

# Directory to store the HBase logs.
HBASE_PID_DIR="/var/run/hbase";

#
# Hadoop Service - ZooKeeper
#

# Directory where ZooKeeper will store data. For example, /grid1/hadoop/
zookeeper/data
ZOOKEEPER_DATA_DIR="../hadoop/zookeeper/data";

# Directory to store the ZooKeeper configuration files.
ZOOKEEPER_CONF_DIR="/etc/zookeeper/conf";

# Directory to store the ZooKeeper logs.
```

```
ZOOKEEPER_LOG_DIR="/var/log/zookeeper";

# Directory to store the ZooKeeper process ID.
ZOOKEEPER_PID_DIR="/var/run/zookeeper";

#
# Hadoop Service - Pig
#

# Directory to store the Pig configuration files.
PIG_CONF_DIR="/etc/pig/conf";

# Directory to store the Pig logs.
PIG_LOG_DIR="/var/log/pig";

# Directory to store the Pig process ID.
PIG_PID_DIR="/var/run/pig";


#
# Hadoop Service - Oozie
#

# Directory to store the Oozie configuration files.
OOZIE_CONF_DIR="/etc/oozie/conf"

# Directory to store the Oozie data.
OOZIE_DATA="/var/db/oozie"

# Directory to store the Oozie logs.
OOZIE_LOG_DIR="/var/log/oozie"

# Directory to store the Oozie process ID.
OOZIE_PID_DIR="/var/run/oozie"

# Directory to store the Oozie temporary files.
OOZIE_TMP_DIR="/var/tmp/oozie"

#
# Hadoop Service - Sqoop
#
SQOOP_CONF_DIR="/etc/sqoop/conf"

export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec
```

Define users and groups:

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

Note

The `scripts.zip` file you downloaded in Download Companion Files includes a script, `usersAndGroups.sh`, for setting user and group environment parameters.

We strongly suggest you edit and source (alternatively, you can also copy the contents to your `~/.bash_profile`) to set up these environment variables in your environment.

Table 1.3. 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`.
FALCON_USER	User owning the Falcon services. For example, `falcon`.
SQOOP_USER	User owing the Sqoop services. For example, `sqoop`.
HADOOP_GROUP	A common group shared by services. For example, `hadoop`.
KNOX_USER	User owning the Knox Gateway services. For example `knox`.

1.8. [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.

To create these accounts manually, you must:

1. Create the username.

```
hdfs fs -mkdir /user/<username>
```

2. Give that account ownership over its directory.

```
hdfs fs -chown <username> /user/<username>
```

Table 1.4. Typical System Users and Groups

Hadoop Service	User	Group
HDFS	hdfs	hadoop
YARN	yarn	hadoop
MapReduce	mapred	hadoop, mapred
Hive	hive	hadoop
HCatalog/WebHCatalog	hcat	hadoop

Hadoop Service	User	Group
HBase	hbase	hadoop
Falcon	falcon	hadoop
Sqoop	sqoop	hadoop
ZooKeeper	zookeeper	hadoop
Oozie	oozie	hadoop
Knox Gateway	knox	hadoop

1.9. Determine HDP Memory Configuration Settings

Two methods can be used to determine YARN and MapReduce memory configuration settings:

- Use the HDP Utility Script to calculate memory configuration settings
- Manually calculate YARN and MapReduce memory configuration settings

The HDP utility script is the recommended method for calculating HDP memory configuration settings, but information about manually calculating YARN and MapReduce memory configuration settings is also provided for reference.

1.9.1. Use the HDP Utility Script to Calculate Memory Configuration Settings

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

Running the Script

To run the `hdp-configuration-utils.py` script, execute the following command from the folder containing the script:

```
python hdp-configuration-utils.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 hdp-configuration-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=49152MB reserved=16GB usableMem=48GB disks=4
Num Container=8
Container Ram=6144MB
Used Ram=48GB
Unused Ram=16GB
yarn.scheduler.minimum-allocation-mb=6144
yarn.scheduler.maximum-allocation-mb=49152
yarn.nodemanager.resource.memory-mb=49152
mapreduce.map.memory.mb=6144
mapreduce.map.java.opts=-Xmx4096m
mapreduce.reduce.memory.mb=6144
mapreduce.reduce.java.opts=-Xmx4096m
yarn.app.mapreduce.am.resource.mb=6144
yarn.app.mapreduce.am.command-opts=-Xmx4096m
mapreduce.task.io.sort.mb=1792
tez.am.resource.memory.mb=6144
tez.am.java.opts=-Xmx4096m
hive.tez.container.size=6144
hive.tez.java.opts=-Xmx4096m
hive.auto.convert.join.noconditionaltask.size=1342177000
```

1.9.2. Manually Calculate YARN and MapReduce Memory Configuration Settings

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

YARN takes into account all of the available compute 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 = min (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 = max(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 = min (2*12, 1.8* 12, (48-6)/2) = min (24, 21.6, 21) = 21

RAM-per-container = max (2, (48-6)/21) = max (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 = min (2*12, 1.8* 12, (48-6-8)/2) = min (24, 21.6, 17) = 17

RAM-per-container = max (2, (48-6-8)/17) = max (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

Notes:

1. Changing `yarn.scheduler.minimum-allocation-mb` without also changing `yarn.nodemanager.resource.memory-mb`, or changing `yarn.nodemanager.resource.memory-mb` without also changing `yarn.scheduler.minimum-allocation-mb` changes the number of containers per node.

2. If your installation has high RAM but not many disks/cores, you can free up RAM for other tasks by lowering both `yarn.scheduler.minimum-allocation-mb` and `yarn.nodemanager.resource.memory-mb`.

1.9.2.1. Configuring MapReduce Memory Settings on YARN

MapReduce runs on top of YARN and utilizes YARN Containers to schedule and execute its Map and Reduce tasks. When configuring MapReduce resource utilization on YARN, there are three aspects to consider:

- The physical RAM limit for each Map and Reduce task.
- The JVM heap size limit for each task.
- The amount of virtual memory each task will receive.

You can define a maximum amount of memory for each Map and Reduce task. Since each Map and Reduce task will run in a separate Container, these maximum memory settings should be equal to or greater than the YARN minimum Container allocation.

For the example cluster used in the previous section (48 GB RAM, 12 disks, and 12 cores), the minimum RAM for a Container (yarn.scheduler.minimum-allocation-mb) = 2 GB. Therefore we will assign 4 GB for Map task Containers, and 8 GB for Reduce task Containers.

In `mapred-site.xml`:

```
<name>mapreduce.map.memory.mb</name>
<value>4096</value>
<name>mapreduce.reduce.memory.mb</name>
<value>8192</value>
```

Each Container will run JVMs for the Map and Reduce tasks. The JVM heap sizes should be set to values lower than the Map and Reduce Containers, so that they are within the bounds of the Container memory allocated by YARN.

In `mapred-site.xml`:

```
<name>mapreduce.map.java.opts</name>
<value>-Xmx3072m</value>
<name>mapreduce.reduce.java.opts</name>
```

```
<value>-Xmx6144m</value>
```

The preceding settings configure the upper limit of the physical RAM that Map and Reduce tasks will use. The virtual memory (physical + paged memory) upper limit for each Map and Reduce task is determined by the virtual memory ratio each YARN Container is allowed. This ratio is set with the following configuration property, with a default value of 2.1:

In `yarn-site.xml`:

```
<name>yarn.nodemanager.vmem-pmem-ratio</name>
<value>2.1</value>
```

With the preceding settings on our example cluster, each Map task will receive the following memory allocations:

- Total physical RAM allocated = 4 GB
- JVM heap space upper limit within the Map task Container = 3 GB
- Virtual memory upper limit = 4*2.1 = 8.2 GB

With MapReduce on YARN, there are no longer pre-configured static slots for Map and Reduce tasks. The entire cluster is available for dynamic resource allocation of Map and Reduce tasks as needed by each job. In our example cluster, with the above configurations, YARN will be able to allocate up to 10 Mappers (40/4) or 5 Reducers (40/8) on each node (or some other combination of Mappers and Reducers within the 40 GB per node limit).

1.10. 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
```

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
```

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
```

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 process ID.

  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.scheduler.class</name>
 <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.
CapacityScheduler</value>
</property>
```

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

```
<property>
 <name>yarn.nodemanager.log-dirs</name>
 <value>/grid/hadoop/yarn/log</value>
 <description>Use the list of directories from $YARN_LOCAL_LOG_DIR.
               For example, /grid/hadoop/yarn/log,/grid1/hadoop/yarn/
log,/grid2/hadoop/yarn/log</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 the 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. Create a $CLIENT_USER in all the nodes and add it in users group.

```
useradd client
usermod -a -G users client
```

3. As the HDFS user, create a `/user/$CLIENT_USER`.

```
sudo su $HDFS_USER
hdfs dfs -mkdir /user/$CLIENT_USER
hdfs dfs -chown $CLIENT_USER:$CLIENT_USER /user/$CLIENT_USER
hdfs dfs -chmod -R 755 /user/$CLIENT_USER
```

4. Run the smoke test as the $CLIENT_USER, using Terasort and sort 10GB of data.

```
su $CLIENT_USER
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-*.jar teragen 10000 /tmp/teragenout
/usr/lib/hadoop/bin/hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-
examples-*.jar terasort /tmp/teragenout /tmp/terasortout
```

5. Installing Zookeeper

This section describes installing and testing Apache ZooKeeper, a centralized tool for providing services to highly distributed systems.

5.1. Install the ZooKeeper RPMs

In a terminal window, type:

- For RHEL/CentOS/Oracle Linux

```
yum install zookeeper
```

- For SLES

```
zypper install zookeeper
```

5.2. Set Directories and Permissions

Create directories and configure ownership and 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 $ZOO_LOG_DIR;
chown -R $ZOOKEEPER_USER:$HADOOP_GROUP $ZOO_LOG_DIR;
chmod -R 755 $ZOO_LOG_DIR;
```

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

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

   where:

   - *$ZOOKEEPER_USER* is the user owning the ZooKeeper services. For example, `zookeeper`.
   - *$ZOO_LOG_DIR* is the directory to store the ZooKeeper logs. For example, `/var/log/zookeeper`.
   - *$ZOOPIDFILE* is the directory to store the ZooKeeper process ID. For example, `/var/run/zookeeper`.

- *$ZOO_DATA_DIR* is the directory where ZooKeeper will store data. For example, `/grid/hadoop/zookeeper/data`.

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 $ZOO_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 ZooKeeper.

- Extract the ZooKeeper configuration files to a temporary directory.

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

- Copy the configuration files

1. On all hosts create the config directory:

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

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

3. Set appropriate permissions:

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

where:

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

5.4. Start ZooKeeper

In order to install and configure HBase and other Hadoop ecosystem components, you must start the ZooKeeper service:

```
su - zookeeper -c 'source /etc/zookeeper/conf/zookeeper-env.sh ; export
ZOOCFGDIR=/etc/zookeeper/conf;/usr/lib/zookeeper/bin/zkServer.sh start >> /
grid/0/var/log/zookeeper/zoo.out 2>&1'
```

6. Installing HBase

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.

6.1. Install the HBase RPMs

In a terminal window, type:

- For RHEL/CentOS/Oracle Linux

```
yum install zookeeper hbase
```

- For SLES

```
zypper install zookeeper hbase
```

6.2. Set Directories and Permissions

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

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

   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`.
   - *$HBASE_USER* is the user owning the HBase services. For example, `hbase`.
   - *$HADOOP_GROUP* is a common group shared by services. For example, `hadoop`.

6.3. Set Up the Configuration Files

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

- Extract the HBase configuration files to a temporary directory.

  The files are located in the `configuration_files/hbase` directory 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.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 the *$HBASE_CONF_DIR*.

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/../
```

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`.

6.4. Validate the Installation

Use these steps to validate your installation.

1. Start HBase.

   a. 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
```

   b. 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`.

2. Smoke Test HBase.

   From a terminal window, enter:

```
su - $HBASE_USER
```

```
hbase shell
```

In the HBase shell, enter the following command:

```
status
```

6.5. Starting the HBase Thrift and REST APIs

Administrators must manually start the Thrift and REST APIs for HBase.

Starting the HBase Thrift API

Run the following command to start the HBase Thrift API:

```
/usr/bin/hbase thrift start
```

Starting the HBase REST API

Run the following command to start the HBase REST API:

```
/usr/lib/hbase/bin/hbase-daemon.sh start rest --infoport 8085
```

7. Installing Phoenix

To install the Phoenix RPM:

1. Install Phoenix on all HBase Region servers:

   - RHEL/CentOS/Oracle Linux:

     ```
     yum install phoenix
     ```

   - For SLES:

     ```
     zypper install phoenix
     ```

2. Link the Phoenix core jar file to the HBase Master and Region Servers:

   ```
   ln -sf /usr/lib/phoenix/lib/phoenix-core-4.0.0.$version.jar /usr/lib/hbase/
   lib/phoenix.jar
   ```

   where *$version* is the five-digit HDP version of the Phoenix jar. For example, 2.1.2.0-471.

3. Add the following configuration to `hbase-site.xml` on all HBase nodes, the Master Server, and all Region Servers:

   ```
   <property>
      <name>hbase.defaults.for.version.skip</name>
      <value>true</value>
   </property>
   <property>
      <name>hbase.regionserver.wal.codec</name>
      <value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</
   value>
   </property>
   ```

4. Restart the HBase Master and Region Servers.

5. Add the phoenix-4.0-client.jar to the classpath of any Phoenix client.

Configuring Phoenix for Security

Phoenix administrators must perform the following additional steps to configure Phoenix to run in a secure Hadoop cluster:

1. Execute the following command to link the HBase configuration file with the Phoenix libraries:

   ```
   ln -sf <HBASE_CONFIG_DIR>/hbase-site.xml <PHOENIX_HOME>/bin/hbase-site.xml
   ```

2. Execute the following command to link the Hadoop configuration file with the Phoenix libraries:

   ```
   ln -sf <HADOOP_CONFIG_DIR>/core-site.xml <PHOENIX_HOME>/bin/core-site.xml
   ```

Note

Phoenix administrators may safely ignore the following warnings when running the `pssql.py` and `sqlline.py` Phoenix scripts in secure mode:

```
14/04/19 00:56:24 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
 applicable
```

```
14/04/19 00:56:24 WARN util.DynamicClassLoader: Failed to identify
 the fs of dir hdfs://<HOSTNAME>:8020/apps/hbase/data/lib, ignored
java.io.IOException: No FileSystem for scheme: hdfs
```

Smoke Testing Phoenix

Execute the following commands to verify your installation on an unsecure cluster:

Note

The command assumes an unsecure cluster with the following values for configuration parameters:

- `hbase.zookeeper.quorum=localhost`
- `hbase.zookeeper.property.clientPort=2181`
- `zookeeper.znode.parent=/hbase-unsecure`

For a secure cluster, the `zookeeper.znode.parent` configuration property requires a different value:

- `zookeeper.znode.parent=/hbase`

```
cd $PHOENIX_HOME/bin
./psql.py localhost:2181:/hbase-unsecure /usr/share/doc/phoenix-4.0.
0.$version/examples/WEB_STAT.sql /usr/share/doc/phoenix-4.0.0.$version/
examples/WEB_STAT.csv
     /usr/share/doc/phoenix-4.0.0.$version/examples/WEB_STAT_QUERIES.sql
```

where *`$version`* is the four-digit HDP version. For example, 2.1.2.0.

Troubleshooting Phoenix

You may encounter a runtime exception similar to the following:

```
Exception in thread "main" java.lang.IllegalAccessError: class com.google.
protobuf.HBaseZeroCopyByteString cannot access its superclass com.google.
protobuf.LiteralByteString
     at java.lang.ClassLoader.defineClass1(Native Method)
     at java.lang.ClassLoader.defineClass(ClassLoader.java:800)
     at java.security.SecureClassLoader.defineClass(SecureClassLoader.
java:142)
```

To resolve this issue, place `hbase-protocol*.jar` immediately after `hbase-site.xml` in the HADOOP_CLASSPATH environment variable, as shown in the following example:

```
HADOOP_CLASSPATH=/path/to/hbase-site.xml:/path/to/hbase-protocol.jar
```

8. 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

8.1. 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
```

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

8.2. 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`.

8.3. 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
```

9. 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

9.1. Install the Hive and HCatalog RPMs

Note

It is recommended that you set the soft and hard limits for number of processes that the Hive user can consume in your server `/etc/security/limits.conf` file as follows.

For non-secured clusters:

```
<hive user ID>     soft     nproc     128
<hive user ID>     hard     nproc     1024
```

For secured clusters:

```
<hive user ID>     soft     nproc     128
<hive user ID>     hard     nproc     32768
```

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
      yum install hive-hcatalog
     ```

   - For SLES:

     ```
     zypper install hive
      zypper install hive-hcatalog
     ```

2. Optional - Download and install the database connector .jar file for your Hive metastore database.

   By default, Hive uses an embedded Derby database for its metastore. However, you can choose to enable a local or remote database for the Hive metastore. Hive supports Derby, MySQL, Oracle, SQL Server, and Postgres. You will need to install the appropriate JDBC connector for your Hive metastore database. Hortonworks recommends using an embedded instance of the Hive Metastore with HiveServer2. An embedded metastore runs in the same process with HiveServer2 rather than as a separate daemon.

   Note

   Administrators who all Hive queries to run as the `hive` system user rather the actual user who submitted the query must set the `hive.server2.enable.doAs` configuration property to `true` and must configure HiveServer2 to use a local Metastore. A value of `true` for this property requires a local Metastore server.

   For example, if you previously installed MySQL, you would use the following steps to install the MySQL JDBC connector:

   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*
   ```

   b. After the install, the MySQL connector .jar file is placed in the `/usr/share/java/` directory. 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.

9.2. Set Directories and Permissions

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

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

9.3. Set Up the Hive/HCatalog Configuration Files

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

1. If you have not already done so, download and extract the HDP companion files.

   A sample hive-site.xml file is included in the `configuration_files/hive` folder in the HDP companion files.

2. Modify the configuration files.

   In the `configuration_files/hive` directory, edit the `hive-site.xml` file and modify the properties based on your environment. Search for TODO in the files for the properties to replace.

   a. Edit the connection properities for your Hive metastore database in `hive-site.xml`:

```
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://TODO-HIVE-METASTORE-DB-SERVER:TODO-HIVE-
METASTORE-DB-PORT/TODO-HIVE-METASTORE-DB-NAME?createDatabaseIfNotExist=
true</value>
    <description>Enter your Hive Metastore Connection URL, for example if
 MySQL: jdbc:mysql://localhost:3306/mysql?createDatabaseIfNotExist=true</
description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>TODO-HIVE-METASTORE-DB-USER-NAME</value>
    <description>Enter your Hive Metastore database user name.</
description>
  </property>

  <property>
   <name>javax.jdo.option.ConnectionPassword</name>
   <value>TODO-HIVE-METASTORE-DB-PASSWORD</value>
   <description>Enter your Hive Metastore database password.</
description>
  </property>

  <property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>TODO-HIVE-METASTORE-DB-CONNECTION-DRIVER-NAME</value>
```

```
  <description>Enter your Hive Metastore Connection Driver Name, for
 example if MySQL: com.mysql.jdbc.Driver</description>
</property>
```

Optional: If you want storage-based authorization for Hive, set the following Hive authorization parameters in the `hive-site.xml` file:

```
<property>
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
</property>

<property>
  <name>hive.security.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.metastore.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.authenticator.manager</name>
  <value>org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator</
value>
</property>
```

Hive also supports SQL standard authorization. See Hive Authorization for more information about Hive authorization models.

For a remote Hive metastore database, use the following `hive-site.xml` property value to set the IP address (or fully-qualified domain name) and port of the metastore host.

```
<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>
```

To enable HiveServer2 for remote Hive clients, assign a value of a single empty space to this property. Hortonworks recommends using an embedded instance of the Hive Metastore with HiveServer2. An embedded metastore runs in the same process with HiveServer2 rather than as a separate daemon. You can also configure HiveServer2 to use an embedded metastore instance from the command line:

```
hive --service hiveserver2 -hiveconf hive.metastore.uris=" "
```

Optional: By default, Hive ensures that column names are unique in query results returned for SELECT statements by prepending column names with a table alias. Administrators who do not want a table alias prefix to table column names can disable this behavior by setting the following configuration property:

```
<property>
```

```
  <name>hive.resultset.use.unique.column.names</name>
  <value>false</value>
</property>
```

Warning

Hortonworks recommends that deployments disable the DataNucleus cache by setting the value of the `datanucleus.cache.level2.type` configuration parameter to `none`. Note that the `datanucleus.cache.level2` configuration parameter is ignored, and assigning a value of `none` to this parameter will not have the desired effect.

Note

You can also use the HDP utility script to fine-tune your configuration settings based on node hardware specifications.

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`.

9.3.1. Configure Hive and HiveServer2 for Tez

The `hive-site.xml` file in the HDP companion files includes the settings for Hive and HiveServer2 for Tez.

If you have already configured the `hive-site.xml`connection properities for your Hive metastore database as described in the previous section, the only remaining task would be to adjust the `hive.tez.container.size` and `hive.tez.java.opts` values as described in the following section. You can also use the HDP utility script to calculate these Tez configuration settings.

9.3.1.1. Hive-on-Tez Configuration Parameters

Apart from the configurations generally recommended for Hive and HiveServer2 and included in the `hive-site.xml` file in the HDP companion files, for a multi-tenant use-

case, only the following configurations are required in the `hive-site.xml` configuration file to configure Hive for use with Tez.

Table 9.1. Hive-Related Configuration Parameters

Configuration Parameter	Description	Default Value
`hive.execution.engine`	This setting determines whether Hive queries will be executed using Tez or MapReduce.	If this value is set to "mr", Hive queries will be executed using MapReduce. If this value is set to "tez", Hive queries will be executed using Tez. All queries executed through HiveServer2 will use the specified `hive.execution.engine` setting.
`hive.tez.container.size`	The memory (in MB) to be used for Tez tasks. If this is not specified (-1), the memory settings from the MapReduce configurations (`mapreduce.map.memory.mb`)will be used by default for map tasks.	`-1`(not specified) If this is not specified, the memory settings from the MapReduce configurations (`mapreduce.map.memory.mb`)will be used by default.
`hive.tez.java.opts`	Java command line options for Tez. If this is not specified, the MapReduce java opts settings (`mapreduce.map.java.opts`) will be used by default for map tasks.	If this is not specified, the MapReduce java opts settings (`mapreduce.map.java.opts`) will be used by default.
`hive.server2.tez.default.queues`	A comma-separated list of queues configured for the cluster.	The default value is an empty string, which prevents execution of all queries. To enable query execution with Tez for HiveServer2, this parameter must configured.
`hive.server2.tez.sessions.per.default.queue`	The number of sessions for each queue named in the `hive.server2.tez.default.queues`.	1 Larger clusters may improve performance of HiveServer2 by increasing this number.
`hive.server2.tez.initialize.default.sessions`	Enables user to use HiveServer2 without enabling Tez for HiveServer2. Users may potentially may want to run queries with Tez without a pool of sessions.	`false`
`hive.server2.enable.doAs`	Required when the queue-related configurations above are used.	`false`

Examples of Hive-Related Configuration Properties:

```
 <property>
   <name>hive.execution.engine</name>
   <value>tez</value>
 </property>
 <property>
   <name>hive.tez.container.size</name>
   <value>-1</value>
   <description>Memory in mb to be used for Tez tasks. If this is not
specified (-1) then the memory settings for map tasks will be used from
mapreduce configuration</description>
 </property>

 <property>
   <name>hive.tez.java.opts</name>
   <value></value>
   <description>Java opts to be specified for Tez tasks. If this is
not specified then java opts for map tasks will be used from mapreduce
configuration</description>
```

```
</property>

<property>
  <name>hive.server2.tez.default.queues</name>
  <value>default</value>
</property>

<property>
  <name>hive.server2.tez.sessions.per.default.queue</name>
  <value>1</value>
</property>

<property>
  <name>hive.server2.tez.initialize.default.sessions</name>
  <value>false</value>
</property>

<property>
  <name>hive.server2.enable.doAs</name>
  <value>false</value>
</property>
```

Note

Users running HiveServer2 in data analytic tools such as Tableau must reconnect to HiveServer2 after switching between the Tez and MapReduce execution engines.

Tip

You can retrieve a list of queues by executing the following command: `hadoop queue -list`.

9.3.1.2. Using Hive-on-Tez with Capacity Scheduler

You can use the `tez.queue.name` property to specify which queue will be used for Hive-on-Tez jobs. You can also set this property in the Hive shell, or in a Hive script. For more details, see "Configuring Tez with the Capacity Scheduler" on this page.

9.4. Create Directories on HDFS

1. Create the 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 the 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 the 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`.

9.5. Validate the Installation

Use the following steps to validate your installation:

1. Initialize the Hive Metastore database schema.

```
$HIVE_HOME/bin/schematool -initSchema -dbType <$databaseType>
```

- The value for $databaseType can be derby, mysql, oracle, mssql, or postgres.
- $HIVE_HOME is by default configured to `usr/lib/hive`

2. Edit hive-site.xml to set the value of datanucleus.autoCreateSchema to false:

```
<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
  <description>Creates necessary schema on a startup if one doesn't exist</
description>
</property>
```

3. Start the Hive Metastore service.

```
su $HIVE_USER
nohup hive --service metastore -hiveconf hive.log.file=hivemetastore.log
 >$HIVE_LOG_DIR/hive.out 2>$HIVE_LOG_DIR/hive.log &
```

Where *$HIVE_LOG_DIR* is the location of the Hive log directory, for example `/var/log/hive/`

Note

You may receive the following error after running the `su $HIVE_USER` command:

```
su hive
This account is currently not available.
```

If you get this error, you may need to reassign the *$HIVE_USER* shell. You can confirm this by looking for the following line in `etc/passwd`:

```
hive:x:493:493:Hive:/var/lib/hive:/sbin/nologin
```

This indicates that the *$HIVE_USER* is assigned to the `sbin/nologin` shell, which blocks access. You can use the following command to assign the *$HIVE_USER* to the `bin/bash` shell.

```
sudo chsh -s /bin/bash hive
```

This command should return the following:

```
Changing shell for hive.
Shell changed.
```

You should then be able to successfully run the `su` *$HIVE_USER* command.

4. 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;
   ```

5. Start HiveServer2.

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

   Where *$HIVE_LOG_DIR* is the location of the Hive log directory, for example `/var/log/hive/`

6. 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*.

10. Installing and Configuring Apache Tez

Apache Tez is the next generation Hadoop query processing framework and execution engine written on top of YARN. Tez is a replacement for the MapReduce application that supports a range of performance features that radically improve query latency in Hadoop. This improvement in query performance enables the use of Hadoop for interactive queries by business analysts and data scientists. With Tez, Hadoop provides both interactive and batch queries.

Prerequisites

Verify that your cluster meets the following pre-requisites before installing Tez:

- Apache Hadoop 2.4
- YARN

Note

Hadoop administrators can also intall Tez using Ambari, which may reduce installation time by automating the installation across all cluster nodes.

Use the following instructions to install and configure Tez:

1. Install the Tez RPM
2. Configure Tez
3. Validate the Tez Installation
4. Enable Tez for Hive Queries
5. Validate Hive-on-Tez Installation
6. Troubleshooting

Note

There are no additional steps required to secure Tez if your cluster is already configured for security.

10.1. Install the Tez RPM

On all client/gateway nodes:

1. Install the Tez RPMs on all client/gateway nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install tez
     ```

   - For SLES:

     ```
     zypper install tez
     ```

2. Execute the following commands to create the Tez configuration directories. These HDFS directories must be publicly accessible.

Note

The *$HDFS_USER* is the user that owns the HDFS service. For example, `hdfs`.

```
su $HDFS_USER
hdfs dfs -mkdir -p /apps/tez
```

3. Execute the following commands from any one of the cluster client nodes to upload the Tez jar files and their dependencies into HDFS:

```
su $HDFS_USER
hdfs dfs -copyFromLocal /usr/lib/tez/* /apps/tez
hdfs dfs -chown -R  hdfs:users /apps/tez
hdfs dfs -chmod 755 /apps
hdfs dfs -chmod 755 /apps/tez
hdfs dfs -chmod 755 /apps/tez/lib/
hdfs dfs -chmod 644 /apps/tez/*.jar
hdfs dfs -chmod 644 /apps/tez/lib/*.jar
```

4. Execute the following command to verify that the files were copied with the preceeding step:

```
hdfs dfs -ls /apps/tez
```

This should return a list of files that looks similar to this:

```
Found 10 items
drwxr-xr-x   - hdfs users          0 2014-06-04 14:07 /apps/tez/conf
drwxr-xr-x   - hdfs users          0 2014-06-04 14:07 /apps/tez/lib
-rwxr-xr-x   3 hdfs users     748252 2014-06-04 14:07 /apps/tez/tez-api-0.4.
0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users      29846 2014-06-04 14:07 /apps/tez/tez-
common-0.4.0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users     980888 2014-06-04 14:07 /apps/tez/tez-dag-0.4.
0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users     242111 2014-06-04 14:07 /apps/tez/tez-
mapreduce-0.4.0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users     195627 2014-06-04 14:07 /apps/tez/tez-
mapreduce-examples-0.4.0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users     110388 2014-06-04 14:07 /apps/tez/tez-runtime-
internals-0.4.0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users     348559 2014-06-04 14:07 /apps/tez/tez-runtime-
library-0.4.0.2.1.2.0-402.jar
-rwxr-xr-x   3 hdfs users       2615 2014-06-04 14:07 /apps/tez/tez-tests-0.
4.0.2.1.2.0-402.jar
```

10.2. Configure Tez

10.2.1. Tez Configuration

Perform the following steps to configure Tez for your Hadoop cluster:

1. Create a `tez-site.xml` configuration file and place it in the `/etc/tez/conf` configuration directory.

   A sample `tez-site.xml` file is included in the `configuration_files/tez` folder in the HDP companion files.

2. Create the *$TEZ_CONF_DIR* environment variable and set it to to the location of the `tez-site.xml` file.

```
export TEZ_CONF_DIR=/etc/tez/conf
```

3. Create the *$TEZ_JARS* environment variable and set it to the location of the Tez jars and their dependencies.

```
export TEZ_JARS=/usr/lib/tez/*:/usr/lib/tez/lib/*
```

Note

Be sure to include the asterisks (*) in the above command.

4. Configure the `tez.lib.uris` property with the HDFS paths containing the Tez jar files in the `tez-site.xml` file.

```
...
<property>
  <name>tez.lib.uris</name>
  <value>${fs.default.name}/apps/tez/,${fs.default.name}/apps/tez/lib/</
value>
</property>
...
```

5. Add *$TEZ_CONF_DIR* and *$TEZ_JARS* to the $HADOOP_CLASSPATH environment variable.

```
export HADOOP_CLASSPATH=$TEZ_CONF_DIR:$TEZ_JARS:$HADOOP_CLASSPATH
```

   Where:

   - *$TEZ_CONF_DIR* is the location of `tez-site.xml`.
   - *$TEZ_JARS* is the location of Tez jars and their dependencies.

10.2.2. Tez Configuration Parameters

Table 10.1. Tez Configuration Parameters

Configuration Parameter	Description	Default Value
`tez.lib.uris`	Location of the Tez jars and their dependencies. Tez applications download required jar files from this location, so it should be public accessible.	N/A
`tez.am.log.level`	Root logging level passed to the Tez Application Master.	`INFO`
`tez.staging-dir`	The staging directory used by Tez when application developers submit DAGs, or Dynamic Acyclic Graphs. Tez creates all temporary files for the DAG job in this directory.	`/tmp/${user.name}/staging`

Configuration Parameter	Description	Default Value
tez.am.resource.memory.mb	The amount of memory in MB that YARN will allocate to the Tez Application Master. The size increases with the size of the DAG.	1536
tez.am.java.opts	Java options for the Tez Application Master process. The value specified for -Xmx value should be less than specified in tez.am.resource.memory.mb, typically 512 MB less to account for non-JVM memory in the process.	-server -Xmx1024m -Djava.net.preferIPv4Stack=true -XX:+UseNUMA -XX:+UseParallelGC
tez.am.shuffle-vertex-manager.min-src-fraction	In case of a Shuffle operation over a Scatter-Gather edge connection, Tez may start data consumer tasks before all the data producer tasks complete in order to overlap the shuffle IO. This parameter specifies the fraction of producer tasks which should complete before the consumer tasks are scheduled. The percentage is expressed as a decimal, so the default value of 0.2 represents 20%.	0.2
tez.am.shuffle-vertex-manager.max-src-fraction	In case of a Shuffle operation over a Scatter-Gather edge connection, Tez may start data consumer tasks before all the data producer tasks complete in order to overlap the shuffle IO. This parameter specifies the fraction of producer tasks which should complete before all consumer tasks are scheduled. The number of consumer tasks ready for scheduling scales linearly between min-fraction and max-fraction. The percentage is expressed as a decimal, so the default value of 0.4 represents 40%.	0.4
tez.am.am-rm.heartbeat.interval-ms.max	This parameter determines how frequently the Tez Application Master asks the YARN Resource Manager for resources in milliseconds. A low value can overload the Resource Manager.	250
tez.am.grouping.split-waves	Specifies the number of *waves*, or the percentage of queue container capacity, used to process a data set where a value of1 represents 100% of container capacity. The Tez Application Master considers this parameter value, the available cluster resources, and the resources required by the application to calculate *parallelism*, or the number of tasks to run. Processing queries with additional containers leads to lower latency. However, resource contention may occur if multiple users run large queries simultaneously.	Tez Default:1.4; Hive Default: 1.7
tez.am.grouping.min-size	Specifies the lower bound of the size of the primary input to each task when The Tez Application Master determines the parallelism of primary input reading tasks. This configuration property prevents input tasks from being too small, which prevents the parallelism for the tasks being too large.	16777216
tez.am.grouping.max-size	Specifies the upper bound of the size of the primary input to each task when the Tez Application Master determines the parallelism of primary input reading tasks. This configuration property prevents input tasks from being too large, which prevents their parallelism from being too small.	1073741824
tez.am.container.reuse.enabled	A *container* is the unit of resource allocation in YARN. This configuration parameter determines whether Tez will reuse the same container to run multiple tasks. Enabling this parameter improves performance by avoiding the memory overhead of reallocating container resources for every task. However, disable this parameter if the tasks contain memory leaks or use static variables.	true

Configuration Parameter	Description	Default Value
tez.am.container.reuse.rack-fallback.enabled	Specifies whether to reuse containers for rack-local tasks. This configuration parameter is ignored unless tez.am.container.reuse.enabled is enabled.	true
tez.am.container.reuse.non-local-fallback.enabled	Specifies whether to reuse containers for non-local tasks. This configuration parameter is ignored unless tez.am.container.reuse.enabled is enabled.	true
tez.am.container.session.delay-allocation-millis	Determines when a Tez session releases its containers while not actively servicing a query. Specify a value of -1 to never release an idle container in a session. However, containers may still be released if they do not meet resource or locality needs. This configuration parameter is ignored unless tez.am.container.reuse.enabled is enabled.	10000 (10 seconds)
tez.am.container.reuse.locality.delay-allocation-millis	The amount of time to wait in milliseconds before assigning a container to the next level of locality. The three levels of locality in ascending order are NODE, RACK, and NON_LOCAL.	250
tez.task.get-task.sleep.interval-ms.max	Determines the maximum amount of time in milliseconds a container agent waits before asking The Tez Application Master for another task. Tez runs an agent on a container in order to remotely launch tasks. A low value may overload the Application Master.	200
tez.session.client.timeout.secs	Specifies the amount of time in seconds to wait for the Application Master to start when trying to submit a DAG from the client in session mode.	180
tez.session.am.dag.submit.timeout.secs	Specifies the amount of time in seconds that the Tez Application Master waits for a DAG to be submitted before shutting down. The value of this property is used when the Tez Application Manager is running in Session mode, which allows multiple DAGs to be submitted for execution. The idle time between DAG submissions should not exceed this time.	300
tez.runtime.intermediate-output.should-compress	Specifies whether Tez should compress intermediate output.	false
tez.runtime.intermediate-output.compress.codec	Specifies the codec to used when compressing intermediate output. This configuration is ignored unless tez.runtime.intermediate-output.should-compress is enabled.	org.apache.hadoop.io.compress.SnappyCo
tez.runtime.intermediate-input.is-compressed	Specifies whether intermediate output is compressed.	false
tez.runtime.intermediate-input.compress.codec	Specifies the codec to use when reading intermediate compressed input. This configuration property is ignored unless tez.runtime.intermediate-input.is-compressed is enabled.	org.apache.hadoop.io.compress.SnappyCo
tez.yarn.ats.enabled	Specifies that Tez should start the TimeClient for sending information to the Timeline Server.	false

10.2.3. Configuring Tez with the Capacity Scheduler

You can use the tez.queue.name property to specify which queue will be used for Tez jobs. Currently the Capacity Scheduler is the default Scheduler in HDP. In general, this is not limited to the Capacity Scheduler, but applies to any YARN queue.

If no queues have been configured, the default queue will be used, which means that 100% of the cluster capacity will be used when running Tez jobs. If queues have been configured, a queue name must be configured for each YARN application.

Setting `tez.queue.name` in `tez-site.xml` would apply to Tez applications that use that configuration file. To assign separate queues for each application, you would need separate tez-site.xml files, or you could have the application pass this configuration to Tez while submitting the Tez DAG.

For example, in Hive you would use the the `tez.queue.name` property in `hive-site.xml` to specify the queue to be used for Hive-on-Tez jobs. To assign Hive-on-Tez jobs to use the "engineering" queue, you would add the following property to `hive-site.xml`:

```
<property>
    <name>tez.queue.name</name>
    <value>engineering</value>
</property>
```

Setting this configuration property in `hive-site.xml` will affect all Hive queries that read that configuration file.

To assign Hive-on-Tez jobs to use the "engineering" queue in a Hive query, you would use the following command in the Hive shell or in a Hive script:

```
set tez.queue.name=engineering;
```

10.3. Validate the Tez Installation

Use the following procedure to run an example Tez application, such as OrderedWordCount, and validate your Tez installation.

1. Create a sample `test.txt` file:

   ```
   foo
   bar
   foo
   bar
   foo
   ```

2. Log in as the *$HDFS_USER*. The *$HDFS_USER* is the user that owns the HDFS service. For example, `hdfs`:

   ```
   su $HDFS_USER
   ```

3. Upload the `test.txt` file into HDFS:

   ```
   hadoop fs -put test.txt /tmp/test.txt
   ```

4. Execute the following command to run the OrderedWordCount application using Tez:

   ```
   hadoop jar /usr/lib/tez/tez-mapreduce-examples-*.jar orderedwordcount /tmp/
   test.txt /tmp/out
   ```

5. Run the following command to verify the word count:

   ```
   hadoop fs -cat '/tmp/out/*'
   ```

This should return:

```
foo 3
bar 2
```

10.4. Enable Tez for Hive Queries

Limitations

This release of Tez does not support the following actions:

- SMB joins
- **SELECT TRANSFORM** queries
- Index creation
- Skew joins

Use the following instructions to enable Tez for Hive Queries:

1. Copy the `hive-exec-0.13.0.jar` to HDFS at the following location: `/apps/hive/install/hive-exec-0.13.0.jar`.

   ```
   su - $HIVE_USER
   hadoop fs -mkdir /apps/hive/install
   hadoop fs -copyFromLocal /usr/lib/hive/lib/hive-exec-* /apps/hive/install/
   hive-exec-0.13.0.jar
   ```

2. Enable Hive to use Tez DAG APIs. On the Hive client machine, add the following to your Hive script or execute it in the Hive shell:

   ```
   set hive.execution.engine=tez;
   ```

Disabling Tez for Hive Queries

Use the following instructions to disable Tez for Hive queries:

On the Hive client machine, add the following to your Hive script or execute it in the Hive shell:

```
set hive.execution.engine=mr;
```

Tez will then be disabled for Hive queries.

10.5. Validate Hive-on-Tez Installation

Use the following procedure to validate your configuration of Hive-on-Tez:

1. Create a sample test.txt file:

   ```
   echo -e "alice miller\t49\t3.15" > student.txt
   ```

2. Upload the new data file to HDFS:

```
su $HDFS_USER
hadoop fs -mkdir -p /user/test/student
hadoop fs -copyFromLocal student.txt /user/test/student
```

3. Open the Hive command-line shell:

```
su $HDFS_USER
hive
```

4. Create a table named `student` in Hive:

```
hive> CREATE EXTERNAL TABLE student(name string, age int, gpa double) ROW
 FORMAT DELIMITED FIELDS TERMINATED BY '\t'
STORED AS TEXTFILE LOCATION '/user/test/student';
```

5. Execute the following query in Hive:

```
hive> SELECT COUNT(*) FROM student;
```

   If Hive-on-Tez is configured properly, this query should successfully return results:

```
hive> SELECT COUNT(*) FROM student;
Query ID = hdfs_20140604161313_544c4455-dfb3-4119-8b08-b70b46fee512
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1401734196960_0007, Tracking URL = http://c6401.ambari.
apache.org:8088/proxy/application_1401734196960_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1401734196960_0007
Hadoop job information for Stage-1: number of mappers: 1; number of
 reducers: 1
2014-06-04 16:13:24,116 Stage-1 map = 0%,  reduce = 0%
2014-06-04 16:13:30,670 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.
82 sec
2014-06-04 16:13:39,065 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU
 1.97 sec
MapReduce Total cumulative CPU time: 1 seconds 970 msec
Ended Job = job_1401734196960_0007
MapReduce Jobs Launched:
Job 0: Map: 1  Reduce: 1   Cumulative CPU: 1.97 sec   HDFS Read: 240 HDFS
 Write: 2 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 970 msec
OK
1
Time taken: 28.47 seconds, Fetched: 1 row(s)
hive>
```

10.6. Troubleshooting

View the Tez logs to help troubleshoot problems with your installation and configuration. Tez logs are accessible through the YARN CLI using the `yarn logs` command.

```
yarn logs -applicationId <APPLICATION_ID> [OPTIONS]
```

You can find the application ID at the end of the output to a running application, as shown in the following output from the OrderedWordCount application.

```
14/02/26 14:45:33 INFO examples.OrderedWordCount: DAG 1 completed. FinalState=
SUCCEEDED
14/02/26 14:45:33 INFO client.TezSession: Shutting down Tez
 Session, sessionName=OrderedWordCountSession, applicationId=
application_1393449467938_0001
```

Apache Tez is still maturing as a technology, and there is not yet complete feature parity between it and Apache MapReduce. Try the following troubleshooting techniques if a Tez-based application does not run successfully.

- Does the application run successfully with Query Vectorization turned off?
- Does the application run successfully with Tez in a different session?
- Does the application run successfully with MapReduce?

  Include all configuration settings and the answers to the above three questions with your support call. Providing Hortonworks Technical Support with the Tez configuration settings used for a problematic query is critical. The same query run with both the Tez and MapReduce execution engines should return the same results.

11. Installing WebHCat

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

Use the following instructions to install WebHCat:

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

11.1. Install the WebHCat RPMs

On the WebHCat server machine, install the necessary RPMs.

- For RHEL/CentOS/Oracle Linux:

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

- For SLES:

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

11.2. Set Directories and Permissions

Create directories and configure ownership and 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`.

11.3. 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-site.xml` and modify the following properties:

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

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

```
<property>
<name>templeton.controller.map.mem</name>
<value>1600</value>
<description>Total virtual memory available to map tasks.</description>
</property>
```

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

3. Set up the WebHCat configuration files.

a. Delete any existing WebHCat configuration files:

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

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

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

where:

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

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

1. Set up the WebHCat user.

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

2. Prepare WebHCat directories on HDFS.

```
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:

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

11.5. Validate the Installation

1. Start the WebHCat server.

```
<login as $WEBHCAT_USER>
su -hcat -c '/usr/lib/hive-hcatalog/sbin/webhcat/_server.sh --config /etc/
hcatalog/conf/webhcat 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"}
```

12. 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

12.1. Install the Oozie RPMs

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
     ```

2. Optional - Enable the Oozie Web Console
   - Create a lib extension directory.

     ```
     cd /usr/lib/oozie
     ```

   - 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/
       ```

   - Add LZO JAR files.

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

3. Optional: Add database connector JAR files.

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

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

   b. Add the downloaded JAR files.

```
/usr/lib/oozie/bin/oozie-setup.sh -hadoop 2.x /usr/lib/hadoop -extjs /
usr/share/HDP-oozie/ext-2.2.zip -jars /usr/lib/hadoop/lib/hadoop-lzo-0.5.
0.jar:/usr/share/java/mysql-connector-java.jar
```

12.2. Set Directories and Permissions

Create directories and configure ownership and 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;
```

```
mkdir /etc/oozie/conf/action-conf
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.

12.3. 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>
```

```
<property>
  <name>oozie.service.WorkflowAppService.system.libpath</name>
  <value>/user/$OOZIE_USER/share/lib</value>
</property>
```

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

```
modify: export JAVA_HOME=
add: JRE_HOME=$
{JAVA_HOME}

export OOZIE_CONFIG=$
{OOZIE_CONFIG:-/etc/oozie/conf}

export CATALINA_BASE=$
{CATALINA_BASE:-/var/lib/oozie/oozie-server}

export CATALINA_TMPDIR=$
{CATALINA_TMPDIR:-/var/tmp/oozie}

export OOZIE_CATALINA_HOME=/usr/lib/bigtop-tomcat
```

3. Copy the Configuration Files

   On your Oozie server create the config directory, copy the configuration 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.

6. Configure your database for Oozie:

   - For MySQL:

```
echo "create database if not exists oozie;" | mysql -u root
echo "grant all privileges on oozie.* to 'oozie'@'localhost' identified by
 'oozie';" | mysql -u root
echo "grant all privileges on oozie.* to 'oozie'@`hostname -f` identified
 by 'oozie';" | mysql -u root
```

- For Postgres:

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

- For Oracle:

```
    bash -l -c 'echo "create user oozie identified by oozie;" | sqlplus
 system/root@`hostname -f`'
bash -l -c 'echo "GRANT SELECT_CATALOG_ROLE TO oozie;" | sqlplus system/
root@`hostname -f`'
bash -l -c 'echo "GRANT CONNECT, RESOURCE TO oozie;" | sqlplus system/
root@`hostname -f`'
```

12.4. Validate the Installation

Use these steps to validate your installation.

1. If you are using a non-Derby database, copy your database connector jar file into `/usr/lib/oozie/libext`.

2. Run the setup script to prepare the Oozie Server.

```
cd /usr/lib/oozie/
bin/oozie-setup.sh prepare-war
chmod 777 -R /var/log/oozie
ln -s /etc/oozie/conf/action-conf /etc/oozie/conf.dist/action-conf
```

3. Create the Oozie DB schema

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

4. Start the Oozie server:

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

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

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

6. Access the Oozie Server with the Oozie client.

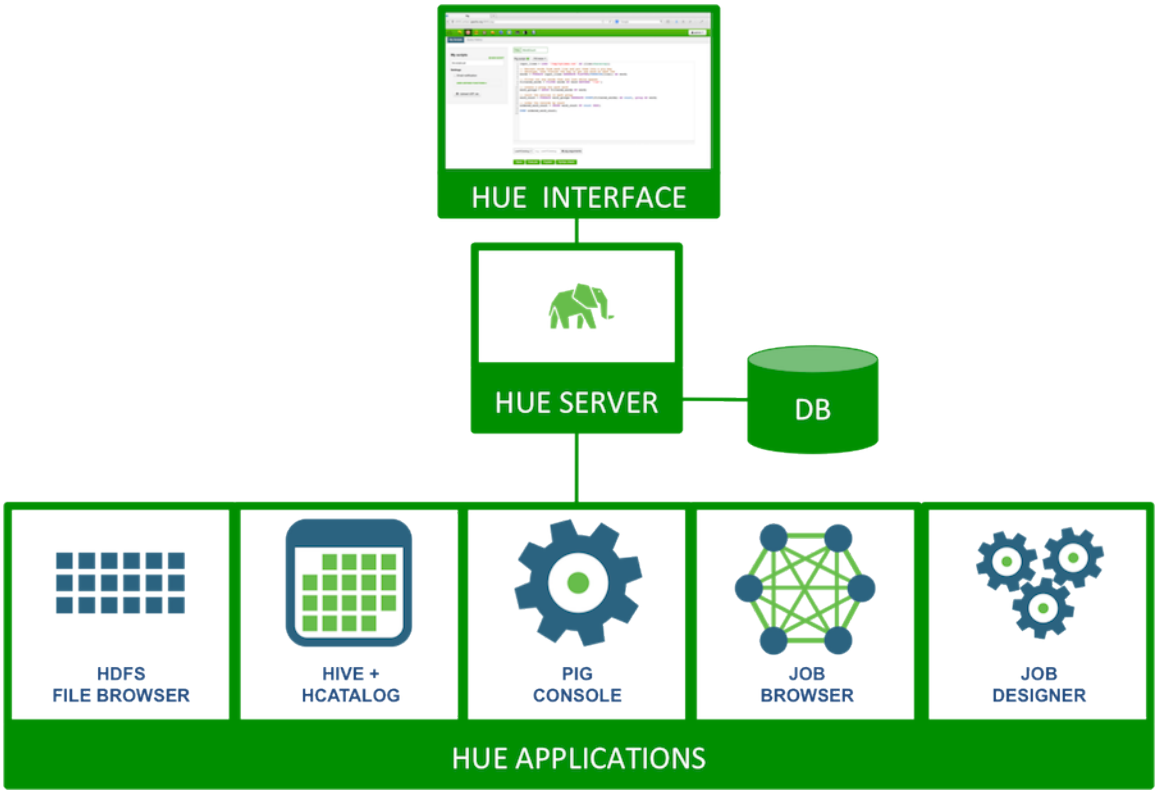
```
oozie admin -oozie http://$oozie.full.hostname:11000/oozie -status
```

You should see the following output:

```
System mode: NORMAL
```

13. 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

13.1. 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 13.1. 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-2.1.X.

   The following table outlines the dependencies on the HDP components:

Table 13.2. 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.

13.2. 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`

13.3. Install Hue

Execute the following command on all Hue Server host machines:

- For RHEL/CentOS/Oracle Linux:

```
yum install hue
```

- For SLES:

```
zypper install hue
```

13.4. 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

13.4.1. 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.

13.4.2. 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`.

13.4.3. 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.

Important

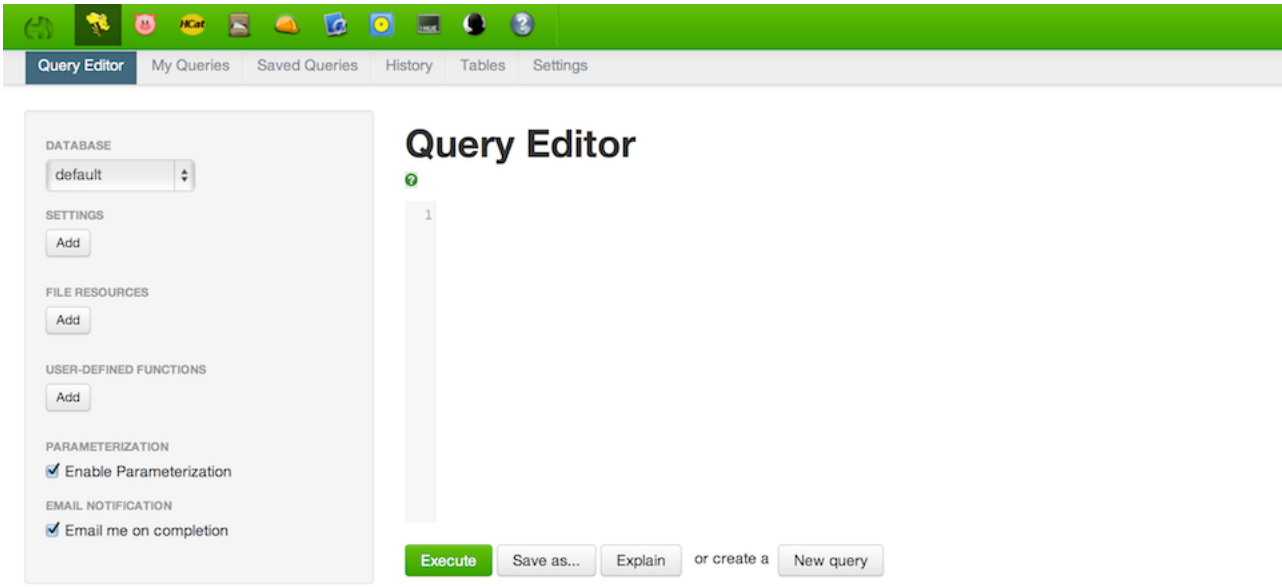
Depending on your environment and the Hive queries you run, queries may fail with an `Internal error processing query` message. Look for an error message `java.lang.OutOfMemoryError: GC overhead limit exceeded` in the `beeswax_server.out` log file. To increase the heap size to avoid this out of memory error, modify the `hadoop-env.sh` file and change the value of `HADOOP_CLIENT_OPTS`.

13.4.3.1. 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.



13.4.4. 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.

13.4.5. 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.

13.4.6. 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. For example: `"http://hostname:50111/templeton/v1/"`

13.5. 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:

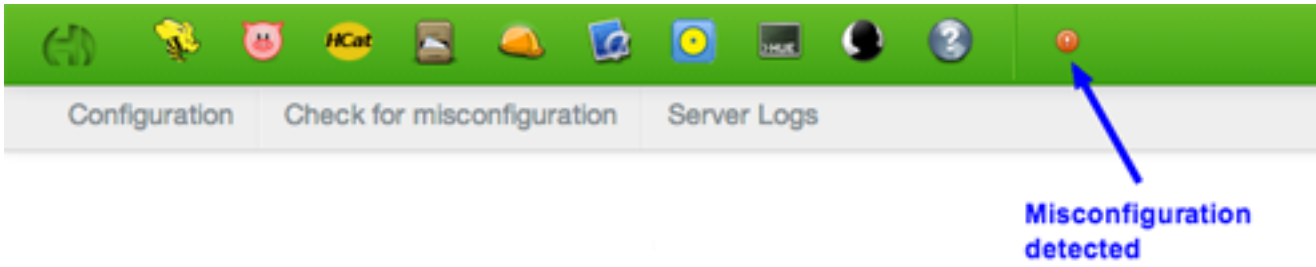
```
/etc/init.d/hue stop
```

To restart Hue, execute the following command:

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

13.6. 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`.

14. 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

14.1. Install the Sqoop RPMs

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
```

14.2. 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 configuration files to the Sqoop configuration directory, such as `/usr/lib/sqoop/conf`.

14.3. Validate the Installation

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

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

15. Installing Mahout

Install 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
```

16. Installing and Configuring 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:

- Understand Flume
- Install Flume
- Configure Flume
- Start Flume
- HDP and Flume
- A Simple Example

16.1. 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.

16.1.1. 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.

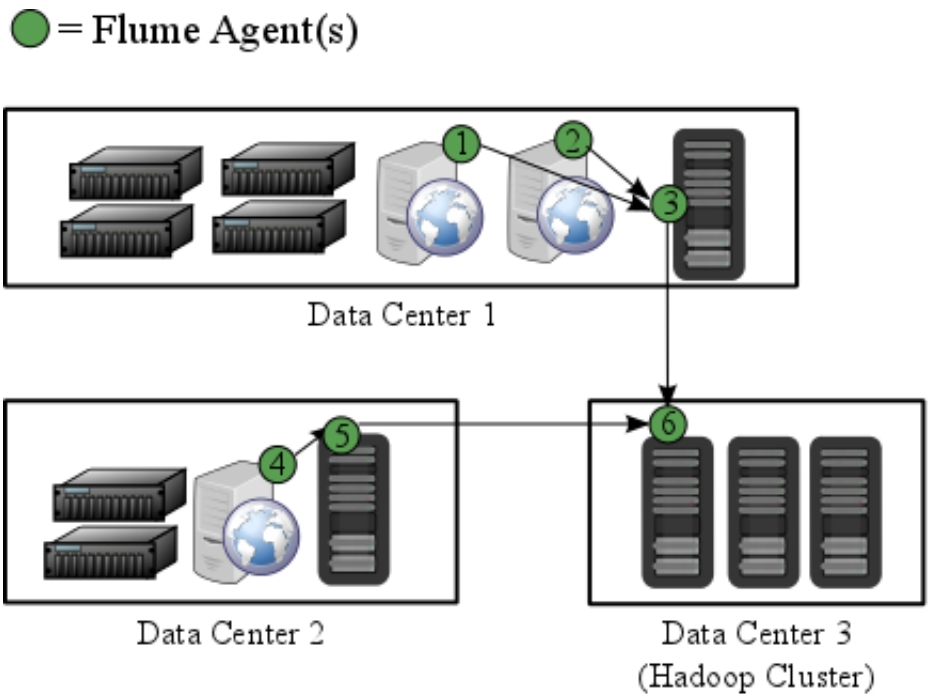
16.2. Install Flume

Flume is included in the HDP repository, but it is not installed automatically as part of the standard HDP installation process. Hortonworks recommends that administrators not install Flume agents on any node in a Hadoop cluster. The following image depicts a sample topology with six Flume agents:

- Agents 1, 2, and 4 installed on web servers in Data Centers 1 and 2.
- Agents 3 and 5 installed on separate hosts in Data Centers 1 and 2 to collect and forward server data in Avro format.
- Agent 6 installed on a separate host on the same network as the Hadoop cluster in Data Center 3 to write all Avro-formatted data to HDFS

Note

It is possible to run multiple Flume agents on the same host. The sample topology represents only one potential data flow.



Note

Hortonworks recommends that administrators use a separate configuration file for each Flume agent. In the diagram above, agents 1, 2, and 4 may have identical configuration files with matching Flume sources, channels, sinks. This is also true of agents 3 and 5. While it is possible to use one large configuration file that specifies all the Flume components needed by all the agents, this is not typical of most production deployments. See Configure Flume for more information about configuring Flume agents.

16.3. Prerequisites

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

   Table 16.1. Flume 1.4.0 Dependencies

Flume 1.4.0 Component	HDP Component Dependencies
HDFS Sink	Hadoop 2.4
HBase Sink	HBase 0.98.0

   See HDP Deployment Options for more information.

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

16.4. Installation

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
```

16.5. Users

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

16.6. Directories

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

16.7. 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-ng/conf/flume.conf`.

To see what configuration properties you can adjust, a template for this file is installed in the configuration directory at: `/etc/flume-ng/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-ng/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
  ```

16.8. Start Flume

There are two options for starting Flume.

- Start Flume directly. On the Flume host:

  ```
  /usr/lib/flume/bin/flume-ng agent -n agent
  ```

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

  ```
  service flume-agent start
  ```

16.9. HDP and Flume

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

16.9.1. Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp

16.9.2. Channels

- Memory
- File

16.9.3. Sinks

- HDFS: secure, nonsecure
- HBase

16.10. 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://<FQDN>:8020/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.

17. Installing and Configuring Apache Storm

This section describes how to install and configure Apache Storm, a distributed, fault-tolerant, and high-performance real time computation tool used to stream data into Hadoop.

Complete the following instructions to install Apache Storm.

1. Install the Storm RPMs.
2. Configure Storm.
3. Configure supervisor to run the Storm daemons.
4. Validate the installation.

17.1. Install the Storm RPMs

Execute the following command on each client cluster node and gateway node to install the Storm RPMs:

- For RHEL/CentOS/Oracle Linux:

```
yum install storm
```

- For SLES:

```
zypper install storm
```

Note

Storm requires version 2.6 or higher as the default system Python interpreter.

17.2. Configure Storm

Use the following procedure to configure Storm:

1. Add the following properties to the `/etc/storm/conf.dist/storm.yaml` file to configure Storm.

```
storm.zookeeper.servers:
    - $ZOOKEEPER_SERVERS

nimbus.host: $NIMBUS_HOSTNAME

drpc.servers:
    - "localhost"
```

```
storm.local.dir: $STORM_LOCAL_DIR

logviewer.port: 8081
```

ZOOKEEPER_SERVERS is a comma-separated list of ZooKeeper servers.

NIMBUS_HOSTNAME is the hostname where the Storm Nimbus server is started.

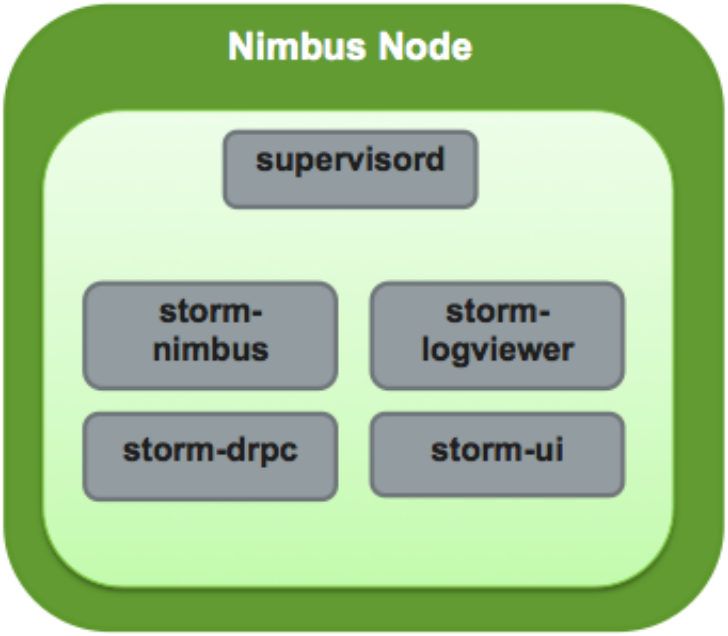
STORM_LOCAL_DIR should be `/tmp/storm/local`, and it must exist on all Storm nodes.

2. Execute the following commands:

```
chown -R storm:storm $STORM_LOCAL_DIR
chmod -R 755 $STORM_LOCAL_DIR
```

17.3. Configure Process Controller

Storm administrators should install and configure a process controller to monitor and run Apache Storm under supervision. Storm is a fail-fast application, meaning that it is designed to fail under certain circumstances, such as a runtime exception or a break in network connectivity. Without a watchdog process, these events can quickly take down an entire Storm cluster in production. A watchdog process prevents this by monitoring for failed Storm processes and restarting them when necessary. This section describes how to configure `supervisord` to manage the Storm processes, but adminstrators may use another process controller of their choice, such as `monit` or `daemontools`.



Add the following stanzas to the `/etc/supervisord.conf` to configure Supervisor to start and stop all the Storm daemons:

```
...
[program:storm-nimbus]
```

```
command=storm nimbus
directory=/home/storm
autorestart=true
user=storm

[program:storm-supervisor]
command=storm supervisor
directory=/home/storm
autorestart=true
user=storm

[program:storm-ui]
command=storm ui
directory=/home/storm
autorestart=true
user=storm

[program:storm-logviewer]
command=storm logviewer
autorestart=true
user=storm

[program:storm-drpc]
command=storm drpc
directory=/home/storm
autorestart=true
user=storm
```

17.4. Validate the Installation

Validate the Apache Storm installation to verify a successful installation and configuration.

Note

You must start ZooKeeper before starting Storm.

1. Execute the following command to start the Storm daemons:

   - RHEL/CentOS/Oracle Linux

     ```
     /usr/bin/supervisord start
     ```

   - SLES

     ```
     /usr/bin/supervisord start
     ```

2. Execute the following command to view the status of the Storm daemons:

   - RHEL/CentOS/Oracle Linux

     ```
     /usr/bin/supervisorctl status
     ```

   - SLES

     ```
     /usr/bin/supervisorctl status
     ```

   You should see output similar to the following:

```
storm-drpc     RUNNING    pid 3368, uptime 0:31:31
storm-logviewer RUNNING    pid 3365, uptime 0:31:31
storm-nimbus   RUNNING    pid 3370, uptime 0:31:31
storm-supervisor RUNNING    pid 8765, uptime 0:00:12
storm-ui       RUNNING    pid 3369, uptime 0:31:31
```

3. Point your browser to the following URL:

   ```
   http://<storm-ui-server>:8080
   ```

   You should see a web page similar to the following:

Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots
0.9.0.1	34m 15s	0	0

Topology summary

Name	Id	Status	Uptime
WordCount	WordCount-2-1391628483	ACTIVE	21d 6h 5m 2

Supervisor summary

Id	Host	Uptime

Nimbus Configuration

Key	Value
dev.zookeeper.path	/tmp/dev
drpc.childopts	-Xmx768
drpc.invocations.port	3773
drpc.port	3772
drpc.queue.size	128
drpc.request.timeout.secs	600
drpc.servers	["localhos
drpc.worker.threads	64

18. Installing Accumulo

Apache Accumulo is a highly scalable structured and distributed key/value store for high performance data storage and retrieval.

1. Install the accumulo RPM
2. Configure accumulo
3. Validate the installation

Note

Accumulo requires HDFS and ZooKeeper to be running before starting. Password-less SSH must be configured between at least the Accumulo master and TabletServer machines. It is recommended that you run Network Time Protocol (NTP) within the cluster to keep node clocks in sync to avoid problems with automatically timestamped data.

18.1. Install the Accumulo RPM

To install the Accumulo RPM use the following command:

- RHEL/CentOS/Oracle Linux:

```
yum install accumulo
```

- For SLES:

```
zypper install accumulo
```

18.2. Configure Accumulo

Accumulo provides example configurations for you to modify. Copy all files from one of the examples folders from /etc/accumulo/conf/examples/ /standalone to /etc/accumulo/conf and:

1. Make an Accumulo data directory:

```
hadoop fs -mkdir -p /user/accumulo/data
```

2. Change permissions on the data directory to grant access to all users:

   hadoop fs -chmod -R 777 /user/accumulo/data

3. Change ownership of the data directory to the $ACCUMULO_USER.

   hadoop fs -chown -R $ACCUMULO_USER:$HDFS_USER /user/accumulo/data

18.3. Validate Accumulo

To validate that Accumulo is set up correctly:

1. Start the Accumulo service:

   a. Initialize Accumulo:

   ```
   /usr/lib/accumulo/bin/accumulo init
   ```

   b. Enter a instance name and password.

   c. Run the Accumulo start-all.sh script:

   ```
   /usr/lib/accumulo/bin/start-all.sh
   ```

2. View the Accumulo native UI:

   ```
   http://<$accumulo-master>:50095
   ```

19. Installing Falcon

Apache Falcon provides a framework for simplifying the development of data management applications in Apache Hadoop. Falcon enables users to automate the movement and processing of data sets. Instead of hard-coding complex data set and pipeline processing capabilities, Hadoop applications can now rely on the Apache Falcon framework for these functions.

1. Install the Falcon RPM
2. Configure Falcon Entites
3. Configuring Oozie for Falcon
4. Configuring Hive for Falcon
5. Validate Falcon

Note

Falcon works with Oozie jobs, Pig scripts, and Hive queries. We recommend that at a minimum you have Oozie and Pig installed to successfully use Falcon for data governance.

19.1. Install the Falcon RPM

To install the Falcon RPM, use the following command:

- RHEL/CentOS/Oracle Linux:

```
yum install falcon
```

- For SLES:

```
zypper install falcon
```

19.2. Configuring Falcon Entities

Falcon provides the following XML configuration files to build your data pipeline:

- **Cluster:** Defines where your data and processes are stored.
- **Feed:** Defines the datasets to be cleaned and processed.
- **Process:** Consumes feeds, invokes processing logic, and produces further feeds.

After you have installed Falcon, edit the example entities in Defining Data Pipelines or create your own based on the Falcon Schemas.

19.3. Configuring Oozie for Falcon

Falcon uses HCatalog for data availability notification when Hive tables are replicated. Make the following configuration changes to Oozie to ensure Hive table replication in Falcon:

1. Stop the Oozie service on all Falcon clusters.

   Execute these commands on the Oozie host machine.

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

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

2. Copy each cluster's hadoop conf directory to a different location. For example, if you have two clusters, copy one to /etc/hadoop/conf-1 and the other to /etc/hadoop/conf-2.

3. For each oozie-site.xml file, modify the `oozie.service.HadoopAccessorService.hadoop.configurations` property, specifying clusters, the RPC ports of the NameNodes, and HostManagers accordingly.

   For example, if Falcon connects to three clusters, specify:

   ```
   <property>
        <name>oozie.service.HadoopAccessorService.hadoop.configurations</name>
        <value>*=/etc/hadoop/
   conf,$NameNode:$rpcPortNN=$hadoopConfDir1,$ResourceManager1:$rpcPortRM=$hadoopConfDir1,$Na
    :$rpcPortNN =$hadoopConfDir3,$ResourceManager3 :$rpcPortRM
    =$hadoopConfDir3</value>
        <description>
            Comma separated AUTHORITY=HADOOP_CONF_DIR, where AUTHORITY is the
    HOST:PORT of
            the Hadoop service (JobTracker, HDFS). The wildcard '*'
    configuration is
            used when there is no exact match for an authority. The
    HADOOP_CONF_DIR contains
            the relevant Hadoop *-site.xml files. If the path is relative is
    looked within
            the Oozie configuration directory; though the path can be absolute
    (i.e. to point
            to Hadoop client conf/ directories in the local filesystem.
        </description>
      </property>
   ```

4. Add the following properties in bold to the `/etc/oozie/conf/oozie-site.xml` file:

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

```
      <name>oozie.service.URIHandlerService.uri.handlers</name>
      <value>org.apache.oozie.dependency.FSURIHandler,org.apache.oozie.
dependency.HCatURIHandler</val
ue>
</property>
<property>
      <name>oozie.services.ext</name>
      <value>org.apache.oozie.service.JMSAccessorService,
org.apache.oozie.service.PartitionDependencyManagerService,
org.apache.oozie.service.HCatAccessorService
</value>
</property>
<!-- Coord EL Functions Properties -->
<property>
      <name>oozie.service.ELService.ext.functions.coord-job-submit-
instances</name>
      <value>now=org.apache.oozie.extensions.OozieELExtensions#ph1_now_echo,
         today=org.apache.oozie.extensions.OozieELExtensions#ph1_today_echo,
         yesterday=org.apache.oozie.extensions.
OozieELExtensions#ph1_yesterday_echo,
         currentMonth=org.apache.oozie.extensions.
OozieELExtensions#ph1_currentMonth_echo,
         lastMonth=org.apache.oozie.extensions.
OozieELExtensions#ph1_lastMonth_echo, currentYear=org.apache.oozie.
extensions.OozieELExtensions#ph1_currentYear_echo,
         lastYear=org.apache.oozie.extensions.
OozieELExtensions#ph1_lastYear_echo,
         formatTime=org.apache.oozie.coord.
CoordELFunctions#ph1_coord_formatTime_echo,
         latest=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_latest_echo,
         future=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_future_echo
      </value>
</property>
<property>
      <name>oozie.service.ELService.ext.functions.coord-action-create-inst</
name>
      <value>
         now=org.apache.oozie.extensions.OozieELExtensions#ph2_now_inst,
         today=org.apache.oozie.extensions.OozieELExtensions#ph2_today_inst,
         yesterday=org.apache.oozie.extensions.
OozieELExtensions#ph2_yesterday_inst,
         currentMonth=org.apache.oozie.extensions.
OozieELExtensions#ph2_currentMonth_inst,
         lastMonth=org.apache.oozie.extensions.
OozieELExtensions#ph2_lastMonth_inst,
         currentYear=org.apache.oozie.extensions.
OozieELExtensions#ph2_currentYear_inst,
         lastYear=org.apache.oozie.extensions.
OozieELExtensions#ph2_lastYear_inst,
         latest=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_latest_echo,
         future=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_future_echo,
         formatTime=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_formatTime,
         user=org.apache.oozie.coord.CoordELFunctions#coord_user
      </value>
</property>
```

```
<property>
      <name>oozie.service.ELService.ext.functions.coord-action-start</name>
      <value>
         now=org.apache.oozie.extensions.OozieELExtensions#ph2_now,
         today=org.apache.oozie.extensions.OozieELExtensions#ph2_today,
         yesterday=org.apache.oozie.extensions.
OozieELExtensions#ph2_yesterday,
         currentMonth=org.apache.oozie.extensions.
OozieELExtensions#ph2_currentMonth,
         lastMonth=org.apache.oozie.extensions.
OozieELExtensions#ph2_lastMonth,
         currentYear=org.apache.oozie.extensions.
OozieELExtensions#ph2_currentYear,
         lastYear=org.apache.oozie.extensions.
OozieELExtensions#ph2_lastYear,
         latest=org.apache.oozie.coord.CoordELFunctions#ph3_coord_latest,
         future=org.apache.oozie.coord.CoordELFunctions#ph3_coord_future,
         dataIn=org.apache.oozie.extensions.OozieELExtensions#ph3_dataIn,
         instanceTime=org.apache.oozie.coord.
CoordELFunctions#ph3_coord_nominalTime,
         dateOffset=org.apache.oozie.coord.
CoordELFunctions#ph3_coord_dateOffset,
         formatTime=org.apache.oozie.coord.
CoordELFunctions#ph3_coord_formatTime,
         user=org.apache.oozie.coord.CoordELFunctions#coord_user
      </value>
</property>
<property>
      <name>oozie.service.ELService.ext.functions.coord-sla-submit</name>
      <value>
         instanceTime=org.apache.oozie.coord.
CoordELFunctions#ph1_coord_nominalTime_echo_fixed,
         user=org.apache.oozie.coord.CoordELFunctions#coord_user
      </value>
</property>
<property>
      <name>oozie.service.ELService.ext.functions.coord-sla-create</name>
      <value>
         instanceTime=org.apache.oozie.coord.
CoordELFunctions#ph2_coord_nominalTime,
         user=org.apache.oozie.coord.CoordELFunctions#coord_user
      </value>
</property>
```

5. Copy the existing Oozie WAR file to `/usr/lib/oozie/oozie.war`.

   This will make sure all existing items in the WAR file are still present after the current update.

   ```
   su root
   cp $CATALINA_BASE/webapps/oozie.war /usr/lib/oozie/oozie.war
   ```

   Where *$CATALINA_BASE* is the path for the Oozie web app. By default, $CATALINA_BASE is `/var/lib/oozie/oozie-server`.

6. Add the Falcon EL extensions to Oozie.

Copy the extension JAR files provided with the Falcon Server to a temporary directory on the Oozie server. For example, if your standalone Falcon Server is on the same machine as your Oozie server, you can just copy the JAR files.

```
mkdir /tmp/falcon-oozie-jars cp
/usr/lib/falcon/oozie/ext/falcon-oozie-el-extension-0.5.0.2.1.5.0-695.jar \
/tmp/falcon-oozie-jars/
```

7. Package the Oozie WAR file.

```
su oozie
cd /usr/lib/oozie/bin
./oozie-setup.sh prepare-war –d /tmp/falcon-oozie-jars
```

8. Start the Oozie service on all Falcon clusters.

   Execute these commands on the Oozie host machine.

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

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

19.4. Configuring Hive for Falcon

Falcon-generated Hive actions require changes to hive-site.xml to pass the right configuration parameters.

Warning

This configuration change lets you work with Hive tables and Oozie workflows, but impacts all Hive actions, including non-Falcon Oozie workflows.

Under the oozie configuration directory (typically `/etc/oozie/conf`), there is a subdirectory called `action-conf`. Under that directory, either create or modify the file hive-site.xml and add the following property:

```
 <property>
   <name>hive.metastore.execute.setugi</name>
   <value>true</value>
</property>
```

After making this change, restart the Oozie service. If you have Oozie configured for HA, perform this configuration change on all Oozie server nodes.

19.5. Configuring for Secure Clusters

If you are using secure clusters, verify that `hadoop.security.auth_to_local` in core-site.xml is consistent across all clusters.

Warning

Inconsistencies in rules for `hadoop.security.auth_to_local` can lead to issues with delegation token renewals.

You must also create the following property definitions in your cluster entity or entities:

```
<properties>
   <property name="dfs.namenode.kerberos.principal" value="nn/$my.
internal@EXAMPLE.COM"/>
   <property name="hive.metastore.kerberos.principal" value="hive/$my.
internal@EXAMPLE.COM"/>
   <property name="hive.metastore.uris" value="thrift://$my.internal:9083"/>
   <property name="hive.metastore.sasl.enabled" value="true"/>
</properties>
```

Note

Replace $my.internal@EXAMPLE.COM and $my.internal with your own values.

19.6. Validate Falcon

To validate Falcon, submit your entities to Falcon:

1. Submit your cluster entity.

   For example, to submit $sampleClusterFile.xml:

   ```
   falcon entity -type cluster -submit -file $yourClusterFile.xml
   ```

2. Submit your dataset or feed entity.

   For example to sumbit $sampleFeedFile.xml:

   ```
   falcon entity -type feed -submit -file $yourFeedFile.xml
   ```

3. Submit your process entity.

   For example, $sampleProcessFile.xml:

   ```
   falcon entity -type process -submit -file $yourProcessFile.xml
   ```

For each entity, you should see the following success message for sumbit:

```
falcon/default/Submit successful ($entity type) $yourEntityFile
```

For example, for a process entity named rawEmailIngestProcess, you would see a successful message as:

```
falcon/default/Submit successful (process) rawEmailIngestProcess
```

20. Installing Knox

Apache Knox Gateway (Apache Knox) is the Web/REST API Gateway solution for Hadoop. It provides a single access point for all of Hadoop resources over REST. It also enables the integration of enterprise identity management solutions and numerous perimeter security features for REST/HTTP access to Hadoop.

Knox can be installed on kerberized and non-kerberized clusters.

Complete the following instructions to install Knox:

1. Install the Knox RPM
2. Validating the Knox Gateway Installation

20.1. Install the Knox RPMs on the Knox server

To install the Knox RPM, run the following command as root:

- RHEL/CentOS/Oracle Linux:

```
sudo yum install knox
```

- For SLES:

```
zypper install knox
```

The installation creates the following:

- `knox` user in `/etc/passwd`
- Knox installation directory: `/usr/lib/knox`, which is referred to as *`$gateway_home`*.
- Knox configuration directory: `/etc/knox/conf`
- Knox log directory: `/var/log/knox`

20.2. Set up and Validate the Knox Gateway Installation

Setting up and validating the Knox Gateway installation requires a fully operational Hadoop Cluster that can be accessed from the gateway. This section explains how to get the gateway up and running and then test access to your existing cluster with the minimal configuration.

Tip

Use the setup in this section for initial gateway testing. For detailed configuration instructions, see the Knox Gateway Administrator Guide

To set up the gateway and test access:

1. Set the master secret:

```
su -l knox -c "$gateway_home/bin/gateway.sh setup"
```

   You are prompted for the master secret, enter the password at the prompts.

2. Start the gateway:

```
su -l knox -c "$gateway_home/bin/gateway.sh start"
Starting Gateway succeeded with PID 1871.
```

   The gateway starts and the PID is stored in `/var/run/knox`.

3. Start the demo LDAP service that contains the guest user account for testing:

```
su -l knox -c "$gateway_home/bin/ldap.sh start"
Starting LDAP succeeded with PID 1965.
```

Note

In a production environment, we recommend using Active Directory or OpenLDAP for authentication. For detailed instructions on configuring Knox Gateway, see Configuring Authentication in the Knox Gateway Administrator Guide.

4. Verify that the gateway and LDAP service are running:

```
su -l knox -c "$gateway_home/bin/gateway.sh status"
Gateway is running with PID 1871.
su -l knox -c "$gateway_home/bin/ldap.sh status"
LDAP is running with PID 1965.
```

5. Confirm access from gateway host to WebHDFS Service host using telnet:

```
telnet $webhdfs_host $webhdfs_port
```

Warning

You must successfully be able to reach the internal cluster service from the gateway before continuing.

6. Update the WebHDFS host information:

   a. Open the *$gateway_home*`/conf/topologies/sandbox.xml` file in an editor, such as vi.

   b. Find service definition for WebHDFS and update it as follows:

```
<service>
   <role>WEBHDFS</role>
   <url>http://$webhdfs_host:$webhdfs_port/webhdfs</url>
</service>
```

   where *$webhdfs_host* and *$webhdfs_port* (default port is 50070) match your environment.

   c. (Optionally) Comment out the Sandbox specific hostmap information:

```
 <!-- REMOVE SANDBOX HOSTMAP PROVIDER
<provider>
    <role>hostmap</role>
    <name>static</name>
    <enabled>false</enabled>
    <param>
      <name>localhost</name><value>sandbox,sandbox.hortonworks.com</
value>
    </param>
</provider>
-->
```

7. (Optionally) Rename the Sandbox Topology Descriptor file to match the name of your cluster:

```
mv $gateway_home/conf/topologies/sandbox.xml $gateway_home/conf/
topologies/cluster-name.xml
```

   The gateway is now configured to allow access to WebHDFS.

8. On an external client that has curl, enter the following command:

```
curl  -k -ssl3 -u guest:guest-password -X GET "https://$gateway_host:8443/
gateway/sandbox/webhdfs/v1/?op=LISTSTATUS"
```

   where:

   - *sandbox* is the name of the cluster topology descriptor file that you created for testing. If you renamed it, then replace *sandbox* in the command above.
   - *$gateway_host* is the Knox Gateway hostname.

   The status is returned.

21. Installing Ganglia

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

21.1. 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
```

21.2. Install the Configuration Files

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

21.2.1. 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`.

21.2.2. Copy the Configuration Files

On each host in the cluster:

1. Grant execute permissions on the following scripts:
   - `/usr/libexec/hdp/ganglia/setupGanglia.sh`
   - `/usr/libexec/hdp/ganglia/startRrdcached.sh`
2. Add the following line to both setupGanglia.sh and startRrdcached.sh:

```
RRDCACHED_BASE_DIR="/var/lib/ganglia/rrds" right after if [ -z "$
            {rrdcachedRunningPid} " ] then
```

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

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

4. Copy the objects files:

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

5. 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/
```

21.2.3. 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
```

21.2.4. 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.

21.2.5. 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.

21.3. Validate the Installation

Use these steps to validate your installation.

21.3.1. Start the Ganglia Server

On the Ganglia server:

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

21.3.2. Start Ganglia Monitoring on All Hosts

On all hosts:

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

21.3.3. Confirm that Ganglia is Running

Browse to the Ganglia server:

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

22. Installing Nagios

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

22.1. 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 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 nagios-www
```

22.2. Install the Configuration Files

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

22.2.1. 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`.

22.2.2. 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
```

22.2.3. 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/
```

22.2.4. 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
```

22.2.5. 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.

22.2.6. 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
```

22.2.7. 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.

22.2.8. 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 22.1. 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 22.2. 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 22.3. 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

22.2.9. 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
   ```

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.`

22.2.10. 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
   ```

22.2.11. Add Templeton Status and Check TCP Wrapper Commands

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

2. Add the following commands:

```
define command{
        command_name    check_templeton_status
        command_line    $USER1$/check_wrapper.sh $USER1$/
check_templeton_status.sh $HOSTADDRESS$ $ARG1$ $ARG2$ $ARG3$ $ARG4$ $ARG5$
 $ARG6$ $ARG7$
       }

define command{
        command_name    check_tcp_wrapper
        command_line    $USER1$/check_wrapper.sh $USER1$/check_tcp -H
 $HOSTADDRESS$ -p $ARG1$ $ARG2$
       }
```

22.3. Validate the Installation

Use these steps to validate your installation.

22.3.1. Validate the Nagios Installation

Validate the installation.

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

22.3.2. Start Nagios and httpd

Start the Nagios server and httpd.

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

22.3.3. Confirm Nagios is Running

Confirm the server is running.

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

This should return:

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

22.3.4. 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#>
```

22.3.5. 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.

22.3.6. 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.

23. Setting Up Security for Manual Installs

This section provides information on enabling security for a manually installed version of HDP.

23.1. Preparing Kerberos

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

23.1.1. Kerberos Overview

To create secure communication among its various components, HDP uses Kerberos. Kerberos is a third-party authentication mechanism, in which users and services that users wish to access rely on a 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.

23.1.2. Installing and Configuring the KDC

To use Kerberos with HDP, either use an existing KDC or install a new one for HDP only. The following gives a very high level description of the installation process. For more information, see RHEL documentation , CentOS documentation, SLES documentation.

1. Install the KDC server:
   - On RHEL, CentOS, or Oracle Linux, run:

     ```
     yum install krb5-server krb5-libs krb5-auth-dialog  krb5-workstation
     ```

   - On SLES, run:

     ```
     zypper install krb5 krb5-server krb5-client
     ```

2. Update the KDC configuration by replacing `EXAMPLE.COM` with your domain and `kerberos.example.com` with the FQDN of the KDC host; the configuration files are located:
   - On RHEL, CentOS, or Oracle Linux:
     - `/etc/krb5.conf`
     - `/var/kerberos/krb5kdc/kdc.conf`.
   - On SLES:
     - `/etc/krb5.conf`
     - `/var/lib/kerberos/krb5kdc/kdc.conf`
3. Copy the updated `krb5.conf` to every cluster node.

23.1.3. Creating the Database and Setting Up the First Administrator

1. Use the utility `kdb5_util` to create the Kerberos database:
   - On RHEL, CentOS, or Oracle Linux:

     ```
     /usr/sbin/kdb5_util create -s
     ```

   - On SLES:

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

Note

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

2. Set up the KDC Access Control List (ACL):

   - On RHEL, CentOS, or Oracle Linux add administrators to `/var/kerberos/krb5kdc/kadm5.acl`.
   - On SLES, add administrators to `/var/lib/kerberos/krb5kdc/kadm5.acl`.

   Note

   For example, the following line grants full access to the database for users with the `admin` extension:

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

3. Restart `kadmin` for the change to take effect.

4. 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"
   ```

   Now this user can create additional principals either on the KDC machine or through the network. The following instruction assume you are using the KDC machine.

5. On the KDC, start Kerberos:

   - On RHEL, CentOS, or Oracle Linux:

     ```
     /sbin/service krb5kdc start
     /sbin/service kadmin start
     ```

   - On SLES:

     ```
     rckrb5kdc start
     rckadmind start
     ```

23.1.4. Creating Service Principals and Keytab Files for HDP

Each service in HDP must have its own principal. As services do not login with a password to acquire their tickets, their principal's authentication credentials are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal.

First you must create the principal, using mandatory naming conventions. Then you must create the keytab file with that principal's information and copy the file to the keytab directory on the appropriate service host.

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

```
kadmin: addprinc -randkey $principal_name/$service-host-FQDN@$hadoop.realm
```

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

Note

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` property in `hdfs-site.xml`, but the NameNodes still have different fully qualified domain names.

Table 23.1. Service Principals

Service	Component	Mandatory Principal Name
HDFS	NameNode	`nn/$FQDN`
HDFS	NameNode HTTP	`HTTP/$FQDN`
HDFS	SecondaryNameNode	`nn/$FQDN`
HDFS	SecondaryNameNode HTTP	`HTTP/$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`

Service	Component	Mandatory Principal Name
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/*$FQDN*
Gateway	Knox	knox/*$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/$datanode-host@$hadoop.realm
```

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 23.2. 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*	nm.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/*$FQDN*	spnego.service.keytab
HBase Master Server	hbase/*$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
Knox Gateway[b]	knox/*$FQDN*	knox.service.keytab

[a]Only required if you are setting up NameNode HA.

[b]Only required if you are using a Knox Gateway.

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.

23.2. Configuring HDP

This section provides information on configuring HDP 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

23.2.1. Configuration Overview

Configuring HDP for Kerberos has two parts:

- Creating a mapping between service principals and UNIX usernames.

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

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

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

- Adding information to three main service configuration files.

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

23.2.2. Creating Mappings Between Principals and UNIX Usernames

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

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

23.2.2.1. Creating Rules

To accommodate 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.

23.2.2.1.1. 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`

23.2.2.1.2. 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`

23.2.2.1.3. 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`

23.2.2.2. 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
```

23.2.3. Adding Security Information to Configuration Files

To enable security on HDP, 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:

```
 export JSVC_HOME=/usr/lib/bigtop-utils
```

23.2.3.1. core-site.xml

To the core-site.xml file on *every* host in your cluster, you must add the following information:

Table 23.3. 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

Property Name	Property Value	Description
		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.</description>
</property>
```

When using the Knox Gateway, add the following to the `core-site.xml` file on the *master nodes* host in your cluster:

Table 23.4. core-site.xml

Property Name	Property Value	Description
hadoop.proxyuser.knox.groups	users	Grants proxy privileges for knox user.
hadoop.proxyuser.knox.hosts	*$knox_host_FQDN*	Identifies the Knox Gateway host.

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.</description>
</property>
<property>
    <name>hadoop.proxyuser.knox.groups</name>
    <value>users</value>
</property>
<property>
    <name>hadoop.proxyuser.knox.hosts</name>
    <value>Knox.EXAMPLE.COM</value>
</property>
```

23.2.3.2. hdfs-site.xml

To the `hdfs-site.xml` file on *every* host in your cluster, you must add the following information:

Table 23.5. 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.

Property Name	Property Value	Description
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 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
```

23.2.3.3. mapred-site.xml

To the `mapred-site.xml` file on *every* host in your cluster, you must add the following information:

Table 23.6. 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.	

Property Name	Property Value	Description	Final
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 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		
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>
```

23.2.3.4. 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 23.7. 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 _HOST 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/_HOST@EXAMPLE.COM	The Kerberos principal name that should be used to run the HRegionServer process. If _HOST 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 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.MasterOb 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>
```

23.2.3.5. hive-site.xml

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

Table 23.8. 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 _HOST is used as the hostname portion, it will be replaced with the actual hostname of the running instance.
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>
```

23.2.3.5.1. oozie-site.xml

To the `oozie-site.xml` file, you must add the following information:

Table 23.9. 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.
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.
oozie.service.ProxyUserService.proxyuser.knox.groups	users	Grant proxy privileges to the knox user. Note only required when using a Knox Gateway.
oozie.service.ProxyUserService.proxyuser.knox.hosts	$knox_host_FQDN	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

23.2.3.5.2. webhcat-site.xml

To the `webhcat-site.xml` file, you must add the following information:

Table 23.10. 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	

Property Name	Property Value	Description
hadoop.proxyuser.knox.groups	users	Grant proxy privileges to the knox user. Note only required when using a Knox Gateway.
hadoop.proxyuser.knox.hosts	*$knox_host_FQDN*	Identifies the Knox Gateway. Note only required when using a Knox Gateway.

23.3. 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)

23.3.1. 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.

23.3.2. 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`.

23.3.3. 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
```

23.3.4. 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>
```

23.3.5. 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.

23.3.6. 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.

23.3.7. Configure HBase for Access Control Lists (ACL)

Use the following instructions to configure HBase for ACL:

1. Open `kinit` as HBase user.

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

   b. Open `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').

23.4. Setting up One-Way Trust with Active Directory

In environments where users from Active Directory (AD) need to access Hadoop Services, set up one-way trust between Hadoop Kerberos realm and the AD (Active Directory) domain.

Warning

Hortonworks recommends setting up one-way trust after fully configuring and testing your Kerberized Hadoop Cluster.

23.4.1. Configure Kerberos Hadoop Realm on the AD DC

Configure the Hadoop realm on the AD DC server and set up the one-way trust.

1. Add the Hadoop Kerberos realm and KDC host to the DC:

```
ksetup /addkdc $hadoop.realm  $KDC-host
```

2. Establish one-way trust between the AD domain and the Hadoop realm:

```
netdom trust  $hadoop.realm  /Domain:$AD.domain  /add  /realm /
passwordt:$trust_password
```

3. (Optional) If Windows clients within the AD domain need to access Hadoop Services, and the domain does not have a search route to find the services in Hadoop realm, run the following command to create a hostmap for Hadoop service host:

```
ksetup /addhosttorealmmap $hadoop-service-host $hadoop.realm
```

Note

Run the above for each *$hadoop-host* that provides services that need to be accessed by Windows clients. For example, Oozie host, WebHCat host, etc.

4. (Optional) define the encryption type:

```
ksetup /SetEncTypeAttr $hadoop.realm $encryption_type
```

Set encryption types based on your security requirements. Mismatching encryption types causes problems.

Note

Run **`ksetup /GetEncTypeAttr $krb_realm`** to list the available encryption types. Verify that the encryption type is configured for the Hadoop realm in the `krb5.conf`.

23.4.2. Configure the AD Domain on the KDC and Hadoop Cluster Hosts

Add the AD domain as a realm to the `krb5.conf` on the Hadoop cluster hosts. Optionally configure encryption types and UDP preferences.

1. Open the `krb5.conf` file with a text editor and make the following changes:

   a. To `libdefaults`, add the following properties:

   Sets the Hadoop realm as default:

```
[libdefaults]
default_domain = $hadoop.realm
```

   Set the encryption type:

```
[libdefaults]
default_tkt_enctypes = $encryption_types
default_tgs_enctypes = $encryption_types
permitted_enctypes = $encryption_types
```

   where the *`$encryption_types`* match the type supported by your environment. For example:

```
default_tkt_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5
 des-cbc-md5 des-cbc-crc
default_tgs_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5
 des-cbc-md5 des-cbc-crc
permitted_enctypes = aes256-cts aes128-cts rc4-hmac arcfour-hmac-md5 des-
cbc-md5 des-cbc-crc
```

   If TCP is open on the KDC and AD Server:

```
[libdefaults]
udp_preference_limit = 1
```

   b. Add a realm for the AD domain:

```
[realms]
$AD.DOMAIN = {
  kdc = $AD-host-FQDN
  admin_server = $AD-host-FQDN
  default_domain = $AD-host-FQDN
```

```
}
```

c. Save the `krb5.conf` the changes to all Hadoop Cluster hosts.

2. Add the trust principal for the AD domain to the Hadoop MIT KDC:

```
kadmin
kadmin:  addprinc krbtgt/$hadoop.realm@$AD.domain
```

This command will prompt you for the trust password, use the same password as the earlier step.

Note

If the encryption type was defined, then use the following command to configure the AD principal:

```
kadmin:  addprinc -e "$encryption_type"  krbtgt/$hadoop.
realm@$AD.domain
```

24. Upgrade from HDP 1.3 to HDP 2.1 Manually

Use the following instructions to upgrade to the latest release of HDP from HDP 1.3:

1. Getting Ready to Upgrade
2. Upgrade Hadoop
3. Migrate the HDP Configurations
4. Create Local Directories
5. Start HDFS
6. Upgrade ZooKeeper
7. Upgrade HBase
8. Upgrade Hive and HCatalog
9. Upgrade Oozie
10. Upgrade WebHCat (Templeton)
11. Upgrade Pig
12. Upgrade Sqoop
13. Upgrade Flume
14. Upgrade Mahout
15. Upgrade Hue
16. Finalize Upgrade
17. Install New HDP 2.1 Services

24.1. 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`

Note

With HDP 2.1, `/etc/hcatalog/conf` is divided into `/etc/hive-hcatalog/conf` and `/etc/hive-webhcat/conf`.You cannot use `/etc/hcatalog/conf` in HDP 2.1.

- `/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.) For example:

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

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

Note

This example is for unsecure clusters. In secure mode, your cluster requires kerberos credentials the HDFS user.

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
hdfs 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
```

```
hdfs 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. Check for HFiles in V1 format. HBase 0.96.0 discontinues support for HFileV1. Before the actual upgrade, install the HBase 0.96 binaries on a separate host using the `hbase-site.xml` configuration file from the running HBase 0.94 binaries. Then, run the following command against the HBase 0.96 binaries 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.

5. Optional. If you are upgrading HBase on a secure cluster, flush the ACL table by running the following HBase shell command as the $HBase_User.

```
flush '_acl_'
```

6. Stop all HDP 1.3 services (including MapReduce) except HDFS:

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

```
service nagios stop
```

b. Stop Ganglia.

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

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

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

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

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

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

where:

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

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

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

where:

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

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

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

This will stop Hive Metastore and HCatalog services.

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

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

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

g. Stop HBase.

i. Execute these commands on all RegionServers:

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

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

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

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

h. Stop MapReduce

i. Execute these commands on all TaskTrackers slaves:

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

ii. Execute these commands on the HistoryServer host machine:

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

iii. Execute theses commands on the node running the JobTracker host machine:

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

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

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

```
su $HDFS_USER
hdfs dfsadmin -safemode enter
hdfs dfsadmin -saveNamespace
```

8. 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`

9. Finalize any prior HDFS upgrade, if you have not done so already.

```
su $HDFS_USER
hdfs dfsadmin -finalizeUpgrade
```

10. 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 24.1. 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`

11. 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 24.2. 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`

12. Stop HDFS

a. Execute these commands on all DataNodes:

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

If you are running a secure cluster, stop the DataNode as root:

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

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

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

c. Execute these commands on the NameNode host machine:

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

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

13. 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.

24.2. Upgrade Hadoop

1. On all nodes, clean the yum repository.
   - For RHEL/CentOS:

     ```
     yum clean all
     ```

   - For SLES:

     ```
     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*
     ```

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/GA/2.1-
latest/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/GA/2.1-
latest/hdp.repo -O /etc/yum.repos.d/hdp.repo
```

- For SLES 11:

```
wget http://public-repo-1.hortonworks.com/HDP/suse11/2.x/GA/2.1-latest/
hdp.repo -O /etc/zypp/repos.d/hdp.repo
```

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

- For RHEL/CentOS/Oracle Linux:

```
yum repolist
```

- For SLES:

```
zypper repos
```

4. Install Hadoop

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade hadoop*
```

- For SLES:

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

5. Install YARN

- For RHEL/CentOS/Oracle Linux:

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

- For SLES:

```
zypper 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.

24.3. Migrate the HDP Configurations

Configurations and configuration file names have changed between HDP 1.3.2 (Hadoop 1.2.x) and HDP 2.1 (Hadoop 2.4). To successfully upgrade to HDP 2.x, back up your current configuration files, download the new HDP 2.1 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`

   Note

   With HDP 2.1, `/etc/hcatalog/conf` is divided into `/etc/hive-hcatalog/conf` and `/etc/hive-webhcat`.You cannot use `/etc/hcatalog/conf` in HDP 2.1.

   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/zookeeper/conf`
2. Edit `/etc/hadoop/conf/core-site.xml` and set `hadoop.rpc.protection` from none to authentication.
3. Copy your `/etc/hcatalog/conf` configurations to `/etc/hive-hcatalog/conf` and `/etc/hive-webhcat` as appropriate.

4. Download the your HDP 2.x companion files from Download Companion Files and migrate your HDP 1.x configuration.

5. 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`.

6. 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.

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

Table 24.3. HDP 1.3.2 Hadoop Core Site (core-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.1 config	HDP 2.1 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

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.1 config	HDP 2.1 config file
topology.script.number.args	core-site.xml	net.topology.script.number.args	core-site.xml

Note

The hadoop.rpc.protection config needs to specify authentication, integrity and/or privacy. No value defaults to authentication, but an invalid value such as "none" causes an error.

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

Table 24.4. HDP 1.3.2 Hadoop Core Site (hdfs-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.1 config	HDP 2.1 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
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

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.1 config	HDP 2.1 config file
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

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

Table 24.5. 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 2.1 config	HDP 2.1 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
mapred.job.reduce.memory.mb	mapred-site.xml	mapreduce.reduce.memory.mb	mapred-site.xml
ecurity.task.umbilical.protocol.acl	mapred-site.xml	security.job.task.protocol.acl	mapred-site.xml

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

Table 24.6. 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 2.1 config	HDP 2.1 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

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

Table 24.7. HDP 1.3.2 Configs and HDP 2.x for hadoop-env.sh

HDP 1.3.2 config	HDP 2.1 config	Description
JAVA_HOME	JAVA_HOME	Java implementation to use

HDP 1.3.2 config	HDP 2.1 config	Description
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.
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.

12. Add the following properties to the `yarn-site.xml` file:

```
<property>
 <name>yarn.resourcemanager.scheduler.class</name>
 <value>org.apache.hadoop.yarn.server.resourcemanager.scheduler.capacity.
CapacityScheduler</value>
</property>
```

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

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

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

```
<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>
```

```
<property>
  <name>mapreduce.shuffle.port</name>
  <value>13562</value>
</property>
```

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

14. For a secure cluster, add the following properties to `mapred-site.xml`:

```

<property>
    <name>mapreduce.jobhistory.principal</name>
    <value>jhs/_PRINCIPAL@$REALM.ACME.COM</value>
    <description>Kerberos principal name for the MapReduce JobHistory
 Server.</description>
</property>
<property>
    <name>mapreduce.jobhistory.keytab</name>
    <value>/etc/security/keytabs/jhs.service.keytab</value>
    <description>Kerberos keytab file for the MapReduce JobHistory Server.</
description>
</property>
```

15. For a secure cluster, you must also update hadoop.security.auth_to_local in `core-site.xml` to include a rule regarding the mapreduce.jobhistory.principal value you set in the previous step.

```
RULE:[2:$1@$0](PRINCIPAL@$REALM.ACME.COM)s/.*/mapred/
```

where PRINCIPAL and REALM are the kerberos principal and realm you specified in mapreduce.jobhistory.principal.

16. Delete any remaining HDP1 properties in the `mapred-site.xml` file.

24.4. Create Local Directories

You must create local directories for YARN on each NodeManager host in your cluster (in HDP-2, the NodeManager replaces the TaskTracker) and set the appropriate permissions for the YARN log directories. If these directories do not exist, you can create them using the instructions on this page.

1. Set the permissions in 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 the permissions of the directories in `yarn.nodemanager.log-dirs`. If these directories do not exist, you can create them using the instructions on this page. 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}
chmod 755 ${YARN_LOG_DIR}

mkdir ${YARN_PID_DIR}
chown yarn:hadoop ${YARN_PID_DIR}
chmod 755 ${YARN_PID_DIR}
```

24.5. 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.

24.5.1. 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
```

24.5.2. 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.

24.5.3. Start YARN/MapReduce Services

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

1. Optional. If you have a secure cluster, you must create the following principals and keytabs for YARN before you start the YARN service:

```
yarn.resourcemanager.principal
yarn.resourcemanager.keytab
yarn.nodemanager.principal
yarn.nodemanager.keytab
yarn.resourcemanager.webapp.spnego-principal
yarn.nodemanager.webapp.spnego-principal
mapreduce.jobhistory.webapp.spnego-principal
yarn.resourcemanager.webapp.spnego-keytab-file
yarn.nodemanager.webapp.spnego-keytab-file
mapreduce.jobhistory.webapp.spnego-keytab-file
```

2. 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
```

3. 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
```

4. 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
```

24.5.4. 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 sucessfully submitted your first MapReduce job in HDP 2.x. The next steps are to upgrade your other components to 2.x.

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.

24.6. Upgrade ZooKeeper

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

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum upgrade zookeeper
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper rm zookeeper
     zypper install 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`.

24.7. Upgrade HBase

1. Start the Zookeeper zkServer and NameNode services.
2. 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:

     Run the following commands:

     ```
     zypper rm hbase
     zypper install hbase
     ```

3. Replace your configuration after upgrading. Replace the HBase template configuration in `/etc/hbase/conf`.
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
   ```

24.8. Upgrade Hive and HCatalog

1. Stop the Hive Metastore, if you have not done so already.

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

   - For RHEL/CentOS:

     ```
     yum upgrade hive
     yum erase hcatalog
     yum install hive-hcatalog
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper up hive
     zypper erase hcatalog
     zypper install hive-hcatalog
     ```

3. Upgrade the Hive Metastore database schema by running the upgrade scripts included in HDP for your metastore database. HDP includes upgrade scripts for the following databases:

   - MySQL
   - Postgres
   - Oracle
   - MS SQL Server
   - Derby

   Upgrading MySQL Hive Metastore

   Run the following scripts as the root user to upgrade a MySQL Hive Metastore from 1.3.x to 2.1.x:

   ```
   > cd /usr/lib/hive/scripts/metastore/upgrade/mysql
   > mysql hive
   mysql> source upgrade-0.11.0-to-0.12.0.mysql.sql
   mysql> source upgrade-0.12.0-to-0.13.0.mysql.sql
   ```

   Upgrading Postgres Hive Metastore

   Run the following scripts as the root user to upgrade a Postgres Hive Metastore from 1.3.x to 2.1.x:

   ```
   cd /usr/lib/hive/scripts/metastore/upgrade/postgres
   psql -d hive -a -f upgrade-0.11.0-to-0.12.0.postgres.sql
   psql -d hive -a -f upgrade-0.12.0-to-0.13.0.postgres.sql
   ```

   Upgrading Oracle Hive Metastore

   Run the following scripts as the root user to upgrade an Oracle Hive Metastore from 1.3.x to 2.1.x:

   ```
   cd /usr/lib/hive/scripts/metastore/upgrade/oracle
   sqlplus
   ```

```
SQL> @upgrade-0.11.0-to-0.12.0.oracle.sql
SQL> @upgrade-0.12.0-to-0.13.0.oracle.sql
```

4. Edit hive-env.sh to point to the new hive-hcatalog.jar:

```
if [ "${HIVE_AUX_JARS_PATH}" != "" ]; then
export HIVE_AUX_JARS_PATH=/usr/lib/hive-hcatalog/share/hcatalog/hive-
hcatalog-core.jar:${HIVE_AUX_JARS_PATH}
else
  export HIVE_AUX_JARS_PATH=/usr/lib/hive-hcatalog/share/hcatalog/hive-
hcatalog-core.jar
fi
```

5. Download and extract the HDP companion files. Copy the `hive-site.xml` file in the `configuration_files/hive` directory of the extracted companion files to the `etc/hive/conf` directory on your Hive host machine. This new version of the `hive-site.xml` file contains new properties for HDP-2.1 features.

6. Edit the `hive-site.xml` file and modify the properties based on your environment. Search for `TODO` in the file for the properties to replace.

   a. Edit the connection properities for your Hive metastore database in `hive-site.xml`:

```
<property>
    <name>javax.jdo.option.ConnectionURL</name>
    <value>jdbc:mysql://TODO-HIVE-METASTORE-DB-SERVER:TODO-HIVE-
METASTORE-DB-PORT/TODO-HIVE-METASTORE-DB-NAME?createDatabaseIfNotExist=
true</value>
    <description>Enter your Hive Metastore Connection URL, for example if
 MySQL: jdbc:mysql://localhost:3306/mysql?createDatabaseIfNotExist=true</
description>
  </property>

  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>TODO-HIVE-METASTORE-DB-USER-NAME</value>
    <description>Enter your Hive Metastore database user name.</
description>
  </property>

  <property>
   <name>javax.jdo.option.ConnectionPassword</name>
   <value>TODO-HIVE-METASTORE-DB-PASSWORD</value>
   <description>Enter your Hive Metastore database password.</
description>
  </property>

  <property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>TODO-HIVE-METASTORE-DB-CONNECTION-DRIVER-NAME</value>
  <description>Enter your Hive Metastore Connection Driver Name, for
 example if MySQL: com.mysql.jdbc.Driver</description>
</property>
```

   Optional: If you want Hive Authorization, set the following Hive authorization parameters in the `hive-site.xml` file:

```
<property>
```

```
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
</property>

<property>
  <name>hive.security.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.metastore.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.authenticator.manager</name>
  <value>org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator</
value>
</property>
```

For a remote Hive metastore database, use the following `hive-site.xml` property value to set the IP address (or fully-qualified domain name) and port of the metastore host. To enable HiveServer2, leave this property value empty.

```
<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>
```

Note

You can also use the HDP utility script to fine-tune your configuration settings based on node hardware specifications.

7. Edit hive-site.xml to set the value of datanucleus.autoCreateSchema to true:

```
<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>true</value>
  <description>Creates necessary schema on a startup if one doesn't exist</
description>
 </property>
```

8. Edit hive-site.xml to update the value of hive.metadata.export.location to the new, joint hive-hcatalog jar (previously hcatalog-core.jar):

```
<property>
   <name>hive.metadata.export.location</name>
   <value>export HIVE_AUX_JARS_PATH=/usr/lib/hive-hcatalog/share/hcatalog/
hive-hcatalog-core.jar</value>
</property>
```

9. Start the Hive Metastore service.

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

10. Start HiveServer2. On the Hive Metastore host machine, execute the following command:

```
sudo su -l $HIVE_USER -c "nohup /usr/lib/hive/bin/hiveserver2 -hiveconf
 hive.metastore.uris=' ' -hiveconf hive.log.file=hs2.log -hiveconf hive.log.
dir=/grid/0/var/log/hive > /grid/0/var/log/hive/hive.out 2> /grid/0/var/log/
hive/hive.log &"
```

where

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

24.9. Upgrade Oozie

1. Before you upgrade, stop any job in a RUNNING or SUSPENDED state before you upgrade.

Note

Proceeding with an Oozie upgrade when you have running or suspended jobs might cause the upgrade to fail or result in data corruption.

   a. Check for running and suspended jobs using the following commands

```
oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
```

   b. If both commands return results showing that there are no jobs matching the criteria, you can proceed with the upgrade:

```
oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
No Jobs match your criteria!
```

   c. If either of the commands returns a list of jobs similar to the following example, you should terminate each job before proceeding:

```
oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
Job ID App Name Status User Group Started Ended
------------------------------------------------------------------------------
0000005-150710151956050-oozie-oozi-W Spark SUSPENDED oozie - 2015-09-16
 16:47 GMT 2015-09-16 16:47 GMT
------------------------------------------------------------------------------
0000004-150710151956050-oozie-oozi-W Pig1 SUSPENDED oozie - 2015-09-16
 16:46 GMT 2015-09-16 16:46 GMT
```

```
------------------------------------------------------------------------------------
0000003-150710151956050-oozie-oozi-W Pig1 SUSPENDED oozie - 2015-09-11
 00:02 GMT 2015-09-11 00:02 GMT
------------------------------------------------------------------------------------

oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
Job ID App Name Status User Group Started Ended
------------------------------------------------------------------------------------
0000007-150710151956050-oozie-oozi-W Spark RUNNING oozie - 2015-09-16
 16:52 GMT 2015-09-16 16:52 GMT
------------------------------------------------------------------------------------
0000006-150710151956050-oozie-oozi-W Fork_Example RUNNING oozie -
 2015-09-16 16:52 GMT 2015-09-16 16:52 GMT
------------------------------------------------------------------------------------
```

d. Terminate each running or suspended job using the following command:

```
oozie job -oozie http://localhost:11000/oozie -kill
 0000007-150710151956050-oozie-oozi-W
```

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

   - For RHEL/CentOS:

     ```
     yum upgrade oozie
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper up oozie
     ```

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

4. 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
   ```

5. Copy the JDBC jar to libext-customer.

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

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

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

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

6. Copy the JDBC jar of your Oozie database to the libext-customer directory. For example, if you are using MySQL, copy your `mysql-connector-java.jar` to `/usr/lib/oozie/libext-customer` and `/usr/lib/oozie/libtools`.

   Note

   HDP 2.1 does not provide the MySQL jar. You can download one from MySQL, when you download mysql-connector-java here.

7. Copy these files to the libext-customer directory

```
cp /usr/lib/hadoop/lib/hadoop-lzo*.jar /usr/lib/oozie/libext-customer
cp /usr/share/HDP-oozie/ext-2.2.zip /usr/lib/oozie/libext-customer/
```

8. 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. Delete any exisitng `/oozie/share` and replace it with the newly extracted files.

```
su -l hdfs -c "hdfs dfs -chown oozie:hadoop /user/oozie"
su -l hdfs -c "hdfs dfs -chmod -R 755 /user/oozie"
```

9. 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
```

10. 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-customer"
```

    Look for console output to indicate success. For example, if you are using MySQL you should see something similar to:

```
INFO: Adding extension: libext-customer/mysql-connector-java.jar
New Oozie WAR file with added 'JARs' at /var/lib/oozie/oozie-server/webapps/
oozie.war
```

11. Replace the content of `/user/oozie/share` in HDFS. On the Oozie server host:

    a. Verify that all the Oozie jobs have been successfully stupped by running the following commands:

```
oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
```

       If either of these commands indicates suspended or running jobs, you should repeat step 1 before proceeding.

    b. 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
```

c. 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"
```

d. 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"
```

12. 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>
```

13. Start Oozie. Run as root.

```
sudo su -l oozie -c "cd /grid/0/var/log/oozie; /usr/lib/oozie/bin/oozie-
start.sh"
```

14. Check processes.

```
ps -ef | grep -i oozie
```

24.10. Upgrade WebHCat (Templeton)

1. Install WebHCat.

   - For RHEL/CentOS:

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

   - For SLES:

     Run the following command:

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

Also see the instructions on manually deploying the WebHCat instance provided here.

2. Copy the appropriate configurations from `/etc/hcatalog/conf` to `/etc/hive-webhcat/conf/`.

3. Copy the new Pig, Hive and Hadoop-streaming jars to HDFS using the path you specified in `./etc/hive-webhcat/conf/` and change ownership to the hcat user with 755 permissions. For example:

```
hdfs dfs-copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /usr/share/HDP-
webhcat/pig.tar.gz
/usr/lib/hadoop-mapreduce/hadoop-streaming.jar hdfs:///apps/webhcat/.
hdfs dfs -chmod -R 755 hdfs:///apps/webhcat/*
hdfs dfs -chown -R hcat hdfs:///apps/webhcat/*
```

4. Replace your WebHCat configuration after upgrading. Copy your modified`/etc/webhcat/conf` from the template to the configuration directory in all your WebHCat hosts.

5. Start WebHCat:

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

6. Smoke test WebHCat. On the WebHCat host machine, execute the following command:

```
 http://$WEBHCAT_HOST_MACHINE:50111/templeton/v1/status
```

If you are using a secure cluster, excecute the following command:

```
curl --negotiate -u:  http://cluster.$PRINCIPAL.$REALM:50111/templeton/v1/
status
{"status":"ok","version":"v1"}[machine@acme]$
```

7. 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`.

24.11. Upgrade Pig

1. On all the Pig clients, execute the following command:

   - For RHEL/CentOS:

     ```
     yum upgrade pig
     ```

   - For SLES:

     Run the following commands:

```
zypper rm pig
zypper install pig
```

2. You must replace `/etc/hive-webhcat/conf/` your configuration after upgrading. Copy `/etc/pig/conf` from the template to the conf directory in pig hosts.

24.12. Upgrade Sqoop

Upgrade Sqoop. On the Sqoop host machine, execute the following command:

- For RHEL/CentOS/Oracle Linux:

```
yum upgrade sqoop
```

- For SLES:

  Run the following commands:

```
zypper rm sqoop
zypper install sqoop
```

- You must replace your configuration after upgrading. Copy `/etc/sqoop/conf` from the template to the conf directory in sqoop hosts.

24.13. Upgrade Flume

Upgrade 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.

24.13.1. 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.

24.14. Upgrade Mahout

Upgrade 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.

- You must replace your configuration after upgrading. Copy `/etc/mahout/conf` from the template to the conf directory in mahout hosts.

24.14.1. 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
```

24.15. Upgrade Hue

For HDP 2.1, you must use Hue 2.3.1. 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.

```
su $HUE_USER
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:

```
su $HUE_USER
cd /var/lib/hue
mv desktop.db desktop.db.old
sqlite3 desktop.db < ~/desktop.bak
```

24.16. Finalize 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:

```
hdfs dfsadmin -finalizeUpgrade
```

24.17. Install New HDP 2.1 Services

Install new HDP 2.1 Services:

- Tez – Apache Tez is a YARN-based processing framework that greatly improves query response times for Hive and other YARN applications.
- Phoenix – Apache Phoenix is a SQL skin over HBase that makes it easier and faster to build HBase applications.
- Storm – A real-time event stream processing platform that provides fixed, continuous, & low latency processing for very high frequency streaming data.
- Falcon – Apache Falcon provides a framework for simplifying the development of data management applications in Apache Hadoop.
- Knox – Apache Knox is the Web/REST API Gateway solution for Hadoop. It provides a single access point for all of Hadoop resources over REST.

25. Upgrade from HDP 2.0 to HDP 2.1 Manually

This document provides instructions on how to upgrade to HDP 2.1 from the HDP 2.0 release.

Use the following instructions to upgrade to the latest release of HDP 2.1 from HDP 2.0:

1. Getting Ready to Upgrade
2. Upgrade Hadoop
3. Start HDFS
4. Upgrade ZooKeeper
5. Upgrade HBase
6. Upgrade Hive and HCatalog
7. Upgrade Oozie
8. Upgrade WebHCat (Templeton)
9. Upgrade Pig
10. Upgrade Sqoop
11. Upgrade Flume
12. Upgrade Mahout
13. Upgrade Hue
14. Finalize Upgrade
15. Install New HDP 2.1 Services

25.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.0 to HDP 2.1 versions and adding the new HDP 2.1 services.

Warning

Rollback to HDP 2.0 requires an update to HDP 2.0. Contact Support for the patch.

The first step is to ensure you keep your HDP 2.0 configurations:

1. Back up the following HDP directories:
   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hive-hcatalog/conf`
   - `/etc/hive-webhcat/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
   hdfs fsck / -files -blocks -locations > 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 storelibext-customer directoryd 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 any prior HDFS upgrade, if you have not done so already.

```
su $HDFS_USER
hdfs dfsadmin -finalizeUpgrade
```

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 25.1. 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 25.2. 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.

25.2. Upgrade Hadoop

1. On all nodes, clean the yum repository.
   - For RHEL/CentOS:

     ```
     yum clean all
     ```

   - For SLES:

     ```
     zypper clean --all
     ```

2. Upgrade Hadoop
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum upgrade hadoop*
     ```

   - For SLES:

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

   - Verify HDP 2.1 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 2.1 installed:

```
hadoop version
```

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

25.3. 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.

25.3.1. Verify HDFS filesystem health

Analyze if the filesystem is healthy.

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

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

2. Run hdfs namespace and report.

- List directories.

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

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

```
hdfs 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
```

25.3.2. 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 all your ResourceManager hosts.

```
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 all your NodeManager 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
```

25.3.3. 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
```

MapReduce works successfully. You can now 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.

25.4. Upgrade ZooKeeper

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

   - For RHEL/CentOS/Oracle Linux:

```
yum upgrade zookeeper
```

- For SLES:

  Run the following commands:

```
zypper up 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 >> $ZOO_LOG_DIR/zoo.out\"
```

   where

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

25.5. Upgrade HBase

1. Start the Zookeeper zkServer and NameNode services.

2. 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:

     Run the following commands:

```
zypper up hbase
```

3. Replace your configuration after upgrading. Replace the HBase template configuration in `/etc/hbase/conf`.

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
```

25.6. Upgrade Hive and HCatalog

1. Stop the Hive Metastore, if you have not done so already.
2. Upgrade Hive and HCatalog. On the Hive and HCatalog host machines, execute the following commands:
   - For RHEL/CentOS:

     ```
     yum upgrade hive
     yum erase hcatalog
     yum install hive-hcatalog
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper rm hive
     zypper erase hcatalog
     zypper install hive-hcatalog
     ```

3. Upgrade the Hive Metastore database schema.

   ```
   $HIVE_HOME/bin/schematool -upgradeSchema -dbType <$databaseType>
   ```

   The value for $databaseType can be derby, mysql, oracle, or postgres.

4. Download and extract the HDP companion files. Copy the `hive-site.xml` file in the `configuration_files/hive` directory of the extracted companion files to the `etc/hive/conf` directory on your Hive host machine. This new version of the `hive-site.xml` file contains new properties for HDP-2.1 features.
5. Edit the `hive-site.xml` file and modify the properties based on your environment. Search for `TODO` in the file for the properties to replace.

   a. Edit the connection properities for your Hive metastore database in `hive-site.xml`:

   ```
   <property>
       <name>javax.jdo.option.ConnectionURL</name>
       <value>jdbc:mysql://TODO-HIVE-METASTORE-DB-SERVER:TODO-HIVE-
   METASTORE-DB-PORT/TODO-HIVE-METASTORE-DB-NAME?createDatabaseIfNotExist=
   true</value>
       <description>Enter your Hive Metastore Connection URL, for example if
    MySQL: jdbc:mysql://localhost:3306/mysql?createDatabaseIfNotExist=true</
   description>
     </property>
   ```

```
  <property>
    <name>javax.jdo.option.ConnectionUserName</name>
    <value>TODO-HIVE-METASTORE-DB-USER-NAME</value>
    <description>Enter your Hive Metastore database user name.</
description>
  </property>

  <property>
   <name>javax.jdo.option.ConnectionPassword</name>
   <value>TODO-HIVE-METASTORE-DB-PASSWORD</value>
   <description>Enter your Hive Metastore database password.</
description>
  </property>

  <property>
  <name>javax.jdo.option.ConnectionDriverName</name>
  <value>TODO-HIVE-METASTORE-DB-CONNECTION-DRIVER-NAME</value>
  <description>Enter your Hive Metastore Connection Driver Name, for
 example if MySQL: com.mysql.jdbc.Driver</description>
</property>
```

Optional: If you want Hive Authorization, set the following Hive authorization parameters in the `hive-site.xml` file:

```
<property>
  <name>hive.security.authorization.enabled</name>
  <value>true</value>
</property>

<property>
  <name>hive.security.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.metastore.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>

<property>
  <name>hive.security.authenticator.manager</name>
  <value>org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator</
value>
</property>
```

For a remote Hive metastore database, use the following `hive-site.xml` property value to set the IP address (or fully-qualified domain name) and port of the metastore host. To enable HiveServer2, leave this property value empty.

```
<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>
```

Note

You can also use the HDP utility script to fine-tune your configuration settings based on node hardware specifications.

6. Edit hive-site.xml to set the value of datanucleus.autoCreateSchema to false:

```
<property>
  <name>datanucleus.autoCreateSchema</name>
  <value>false</value>
  <description>Creates necessary schema on a startup if one doesn't exist.</
description>
</property>
```

7. Start Hive Metastore service.

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

8. Optional. If you want Hive Authorization, add the Hive authorization parameters to hive-site.xml:

```
<property>
  <name>hive.security.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>
<property>
  <name>hive.security.metastore.authorization.manager</name>
  <value>org.apache.hadoop.hive.ql.security.authorization.
StorageBasedAuthorizationProvider</value>
</property>
<property>
  <name>hive.security.authenticator.manager</name>
  <value>org.apache.hadoop.hive.ql.security.ProxyUserAuthenticator</value>
</property>
```

9. 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 &"
```

10. 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`).

25.7. Upgrade Oozie

1. Before you upgrade, stop any job in a RUNNING or SUSPENDED state before you upgrade.

   Note

   Proceeding with an Oozie upgrade when you have running or suspended jobs might cause the upgrade to fail or result in data corruption.

   a. Check for running and suspended jobs using the following commands

   ```
   oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
   oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
   ```

   b. If both commands return results showing that there are no jobs matching the criteria, you can proceed with the upgrade:

   ```
   oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
   No Jobs match your criteria!
   ```

   c. If either of the commands returns a list of jobs similar to the following example, you should terminate each job before proceeding:

   ```
   oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
   Job ID App Name Status User Group Started Ended
   ------------------------------------------------------------------------------------
   0000005-150710151956050-oozie-oozi-W Spark SUSPENDED oozie - 2015-09-16
    16:47 GMT 2015-09-16 16:47 GMT
   ------------------------------------------------------------------------------------
   0000004-150710151956050-oozie-oozi-W Pig1 SUSPENDED oozie - 2015-09-16
    16:46 GMT 2015-09-16 16:46 GMT
   ------------------------------------------------------------------------------------
   0000003-150710151956050-oozie-oozi-W Pig1 SUSPENDED oozie - 2015-09-11
    00:02 GMT 2015-09-11 00:02 GMT
   ------------------------------------------------------------------------------------

   oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
   Job ID App Name Status User Group Started Ended
   ------------------------------------------------------------------------------------
   0000007-150710151956050-oozie-oozi-W Spark RUNNING oozie - 2015-09-16
    16:52 GMT 2015-09-16 16:52 GMT
   ------------------------------------------------------------------------------------
   0000006-150710151956050-oozie-oozi-W Fork_Example RUNNING oozie -
    2015-09-16 16:52 GMT 2015-09-16 16:52 GMT
   ------------------------------------------------------------------------------------
   ```

   d. Terminate each running or suspended job using the following command:

```
oozie job -oozie http://localhost:11000/oozie -kill
 0000007-150710151956050-oozie-oozi-W
```

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

   - For RHEL/CentOS:

     ```
     yum upgrade oozie
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper up oozie
     ```

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

4. 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
   ```

5. Copy the JDBC jar to libext-customer.

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

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

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

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

6. Copy the JDBC jar of your Oozie database to the libext-customer directory. For example, if you are using MySQL, copy your `mysql-connector-java.jar` to `/usr/lib/oozie/libext-customer` and `/usr/lib/oozie/libtools`.

   Note

   HDP 2.1 does not provide the MySQL jar. You can download one from MySQL, when you download mysql-connector-java here.

7. Copy these files to the libext-customer directory

   ```
   cp /usr/lib/hadoop/lib/hadoop-lzo*.jar /usr/lib/oozie/libext-customer
   cp /usr/share/HDP-oozie/ext-2.2.zip /usr/lib/oozie/libext-customer/
   ```

8. 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. Delete any exisitng `/oozie/share` and replace it with the newly extracted files.

```
su -l hdfs -c "hdfs dfs -chown oozie:hadoop /user/oozie"
su -l hdfs -c "hdfs dfs -chmod -R 755 /user/oozie"
```

9. 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
```

10. 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-customer"
```

Look for console output to indicate success. For example, if you are using MySQL you should see something similar to:

```
INFO: Adding extension: libext-customer/mysql-connector-java.jar
New Oozie WAR file with added 'JARs' at /var/lib/oozie/oozie-server/webapps/
oozie.war
```

11. Replace the content of `/user/oozie/share` in HDFS. On the Oozie server host:

    a. Verify that all the Oozie jobs have been successfully stupped by running the following commands:

    ```
    oozie jobs -oozie http://localhost:11000/oozie -filter status=SUSPENDED
    oozie jobs -oozie http://localhost:11000/oozie -filter status=RUNNING
    ```

    If either of these commands indicates suspended or running jobs, you should repeat step 1 before proceeding.

    b. 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
    ```

    c. 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"
    ```

    d. 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"
```

12. 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>
```

13. Start Oozie. Run as root.

```
sudo su -l oozie -c "cd /grid/0/var/log/oozie; /usr/lib/oozie/bin/oozie-
start.sh"
```

14. Check processes.

```
ps -ef | grep -i oozie
```

25.8. Upgrade WebHCat (Templeton)

1. Upgrade WebHCat.

   - For RHEL/CentOS:

     ```
     yum upgrade hive-webhcat webhcat-tar-hive webhcat-tar-pig
     ```

   - For SLES:

     Run the following command:

     ```
     zypper up hive-webhcat webhcat-tar-hive webhcat-tar-pig
     ```

   Also see the instructions on manually deploying WebHCat instance provided here.

2. You must replace your configuration after upgrading. Copy `/etc/webhcat/conf` from the template to the conf directory in webhcat hosts.

3. Start WebHCat:

   ```
   sudo su -l $WEBHCAT_USER -c "/usr/lib/hive-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
   ```

   If you are using a secure cluster, excecute the following command:

```
curl --negotiate -u:  http://cluster.$PRINCIPAL.$REALM:50111/templeton/v1/
status
{"status":"ok","version":"v1"}[machine@acme]$
```

25.9. Upgrade Pig

1. On all the Pig clients, execute the following command:

   - For RHEL/CentOS:

     ```
     yum upgrade pig
     ```

   - For SLES:

     Run the following commands:

     ```
     zypper up pig
     ```

2. You must replace your configuration after upgrading. Copy `/etc/pig/conf` from the template to the conf directory in pig hosts.

25.10. Upgrade Sqoop

Upgrade Sqoop. On the Sqoop host machine, execute the following command:

- For RHEL/CentOS/Oracle Linux:

  ```
  yum upgrade sqoop
  ```

- For SLES:

  Run the following commands:

  ```
  zypper up sqoop
  ```

- You must replace your configuration after upgrading. Copy `/etc/sqoop/conf` from the template to the conf directory in sqoop hosts.

25.11. Upgrade Flume

Upgrade 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.

25.11.1. 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.

25.12. Upgrade Mahout

Upgrade 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.

- You must replace your configuration after upgrading. Copy `/etc/mahout/conf` from the template to the conf directory in mahout hosts.

25.12.1. 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
```

25.13. Upgrade Hue

For HDP 2.1, you must use Hue 2.3.1 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.

```
su $HUE_USER
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 hue-common hue-beeswax hue-hcatalog hue-pig hue-oozie
```

- For SLES:

```
zypper up hue hue-common hue-beeswax hue-hcatalog hue-pig hue-oozie
```

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:

```
su $HUE_USER
cd /var/lib/hue
mv desktop.db desktop.db.old
sqlite3 desktop.db < ~/desktop.bak
```

25.14. Finalize 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:

```
hdfs dfsadmin -finalizeUpgrade
```

25.15. Install New HDP 2.1 Services

Install new HDP 2.1 Services:

- Tez – Apache Tez is a YARN-based processing framework that greatly improves query response times for Hive and other YARN applications.
- Phoenix – Apache Phoenix is a SQL skin over HBase that makes it easier and faster to build HBase applications.
- Storm – A real-time event stream processing platform that provides fixed, continuous, & low latency processing for very high frequency streaming data.
- Falcon – Apache Falcon provides a framework for simplifying the development of data management applications in Apache Hadoop.
- Knox – Apache Knox is the Web/REST API Gateway solution for Hadoop. It provides a single access point for all of Hadoop resources over REST.

26. 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\*
     ```

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\*
     ```

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\*
     ```

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\*
     ```

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\*
     ```

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\*
```

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

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

9. If Knox is installed, execute the following command on all the gateway host:

- For RHEL/CentOS/Oracle Linux:

```
yum remove knox\*
```

- SLES:

```
zypper remove knox\*
```

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

```
yum remove hadoop\*
```

11. 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\*
```