Hortonworks Data Platform

Non-Ambari Cluster Installation Guide

(July 21, 2015)

docs.cloudera.com

Hortonworks Data Platform: Non-Ambari Cluster Installation Guide

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 Software Foundation projects that focus on the storage and processing of Big Data, along with operations, security, and governance for the resulting system. This includes Apache Hadoop – which includes MapReduce, Hadoop Distributed File System (HDFS), and Yet Another Resource Negotiator (YARN) – along with Ambari, Falcon, Flume, HBase, Hive, Kafka, Knox, Oozie, Phoenix, Pig, Ranger, Slider, Spark, Sqoop, Storm, Tez, and ZooKeeper. 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:

1.1. Meet Minimum System Requirements

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

1.1.1. Hardware Recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. A complete installation of HDP 2.3 will take up about 2.5 GB of disk space. For more information about HDP hardware recommendations, see the "*HDP Cluster Planning Guide*."

1.1.2. Operating System Requirements

The following operating systems are supported:

- 64-bit CentOS 6
- 64-bit CentOS 7
- 64-bit Red Hat Enterprise Linux (RHEL) 6
- 64-bit Red Hat Enterprise Linux (RHEL) 7
- 64-bit Oracle Linux 6
- 64-bit Oracle Linux 7
- 64-bit SUSE Linux Enterprise Server (SLES) 11, SP3/SP4

- 64-bit Debian 6 - supported in first maintenance release of HDP 2.3
- 64-bit Ubuntu Precise (12.04) - supported in first maintenance release of HDP 2.3
- Windows Server 2008, 2012

1.1.3. Software Requirements

Install the following software on each of your hosts:

- `yum` (for RHEL or CentOS)
- `zypper` (for SLES)
- `php_curl` (for SLES)
- `reposync` (may not be installed by default on all SLES hosts)
- `rpm` (for RHEL, CentOS, or SLES)
- `scp`
- `curl`
- `wget`
- `unzip`
- `tar`

In addition, if you are creating local mirror repositories as part of the installation process (see Deploying HDP in Production Data Centers with Firewalls, and you are using RHEL/CentOS/SLES, then on the mirror repo server you will need:

- `yum-utils`
- `createrepo`
- `reposync`

1.1.4. JDK Requirements

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

- Oracle JDK 1.7 64-bit update 67 or higher
- Oracle JDK 1.8 64-bit update 40 or higher
- OpenJDK 1.8 64-bit update 51 or higher
- OpenJDK 1.7 64-bit (latest stable version recommended)

Important

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on all hosts in the cluster. See Installing the JCE for more information.

The following sections describe how to install and configure the JDK.

1.1.4.1. Oracle JDK 1.7 or 1.8

Use the following instructions to manually install JDK 1.7 or JDK 1.8:

1. Verify that you have a `/usr/java` directory. If not, create one:

   ```
   mkdir /usr/java
   ```

2. Download the Oracle 64-bit JDK (jdk-7u67-linux-x64.tar.gz or jdk-8u51-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 or http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html.

3. Copy the downloaded `jdk.tar.gz` file to the `/usr/java` directory.

4. Navigate to the `/usr/java` directory and extract the jdk.tar.gz file.

   ```
   cd /usr/java && tar zxvf jdk-7u67-linux-x64.tar.gz
                 or
   cd /usr/java  tar zxvf jdk-8u51-linux-x64.tar.gz
   ```

   The JDK files will be extracted into a `/usr/java/jdk1.7.0_67` directory or a `/usr/java/jdk1.8.0_51` directory.

5. Create a symbolic link (symlink) to the JDK:

   ```
   ln -s /usr/java/jdk1.7.0_67 /usr/java/default
   ```

   or

   ```
   ln -s /usr/java/jdk1.8.0_51 /usr/java/default
   ```

6. Set the JAVA_HOME and PATH environment variables.

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

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

   ```
   java -version
   ```

   You should see output similar to the following:

   ```

   java version "1.7.0_67"
   Java(TM) SE Runtime Environment (build 1.7.0_67-b01)
   Java HotSpot(TM) 64-Bit Server VM (build 24.67-b01, mixed mode)
   ```

1.1.4.2. OpenJDK 1.7

OpenJDK7 on HDP 2.3 does not work if you are using SLES as your OS. Use the following instructions to manually install OpenJDK 1.7 on a Linux OS other than SLES:

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. For example, if you are using Centos:

   ```
   rpm -qa | grep java
   ```

   ```
   yum remove {java-1.*}
   ```

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

   ```
   which java
   ```

4. (Optional) Download OpenJDK 1.7 RPMs. From the command-line, run:

   RedHat/CentOS/Oracle Linux:

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

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

   ```
   mkdir /usr/java
   ```

   ```
   ln -s /usr/hdp/current/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. (Optional) 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. (Optional) 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)
   ```

1.1.4.3. Installing the JCE

Before enabling Kerberos in the cluster, you must deploy the Java Cryptography Extension (JCE) security policy files on all hosts in the cluster.

Important

If you are using Oracle JDK, you must distribute and install the JCE on all hosts. If you are using OpenJDK, some distributions of the OpenJDK come with unlimited strength JCE automatically and therefore, installation of JCE is not required.

1. Obtain the JCE policy file appropriate for the JDK version in your cluster.

   - For Oracle JDK 1.8:

     http://www.oracle.com/technetwork/java/javase/downloads/jce8-download-2133166.html

   - For Oracle JDK 1.7:

     http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html

2. Save the policy file archive in a temporary location.

3. On each host in the cluster, add the unlimited security policy JCE jars to `$JAVA_HOME/jre/lib/security/`.

   For example, run the following to extract the policy jars into the JDK installed on your host:

   ```
   unzip -o -j -q jce_policy-8.zip -d /usr/jdk64/jdk1.8.0_60/jre/lib/security/
   ```

1.1.5. 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, 9.3+
- MySQL 5.6
- Oracle 11g r2
- SQL Server 2008 R2+

The following sections describe how to install and configure the Metastore database.

1.1.5.1. Metastore Database Prerequisites

The database administrator must create the following users and specify the following values:

- For Hive: hive_dbname, hive_dbuser, and hive_dbpasswd.
- For Oozie: 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.

1.1.5.2. Installing and Configuring PostgreSQL

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

1.1.5.2.1. RHEL/CentOS/Oracle Linux

To install a new instance of PostgreSQL:

1. Connect to the host machine where you plan to deploy PostgreSQL instance.

   At a terminal window, enter:

   ```
   yum install postgresql-server
   ```

2. Start the instance.

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

   Note

   For some newer versions of PostgreSQL, you might need to execute the command: `/etc/init.d/postgresql initdb`

3. Reconfigure PostgreSQL server:

   - Edit the `/var/lib/pgsql/data/postgresql.conf` file.

     Change the value of `#listen_addresses = 'localhost'` to `listen_addresses = '*'`

   - Edit the `/var/lib/pgsql/data/postgresql.conf` file.

     Uncomment the "port = " line and specify the port number (default is 5432)

   - Edit the `/var/lib/pgsql/data/pg_hba.conf` file

     Add the following:

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

   - 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.

   Log in as the root user and enter:

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

Where:

- $postgres is the postgres user
- $user is the user you want to create
- $dbname is the name of your PostgreSQL database

Note

For access to the Hive metastore, create hive_dbuser after Hive has been installed, and for access to the Oozie metastore, create oozie_dbuser after Oozie has been installed.

5. On the Hive Metastore host, install the connector:

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

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

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

1.1.5.2.2. SUSE Linux Enterprise Server (SLES)

To install a new instance of PostgreSQL:

1. Connect to the host machine where you plan to deploy the PostgreSQL instance.

   At a terminal window, enter:

   ```
   zypper install postgresql-server
   ```

2. Start the instance.

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

   Note

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

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

3. Reconfigure the PostgreSQL server:

   - Edit the `/var/lib/pgsql/data/postgresql.conf` file.

     Change the value of `#listen_addresses = 'localhost'` to `listen_addresses = '*'`

   - Edit the `/var/lib/pgsql/data/postgresql.conf` file.

     Change the port setting `#port = 5432` to `port = 5432`

   - Edit the `/var/lib/pgsql/data/pg_hba.conf` file.

Add the following:

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

- **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.

   Log in as the root and enter:

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

   Where:

   - $postgres is the postgres user
   - $user is the user you want to create
   - $dbname is the name of your PostgresSQL database

   Note

   For access to the Hive metastore, create hive_dbuser after Hive has been installed, and for access to the Oozie metastore, create oozie_dbuser after Oozie has been installed.

5. On the Hive Metastore host, install the connector.

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

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

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

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

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

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

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

1.1.5.3. Installing and Configuring MySQL

This section describes how to install MySQL as the metastore database. For instructions on how to install other supported databases, see your third-party documentation.

Important

When you use MySQL as your Hive metastore, you must use `mysql-connector-java-5.1.35.zip` or later JDBC driver.

1.1.5.3.1. RHEL/CentOS

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, enter:

   `yum install mysql-server` (for CentOS6)

   `yum install mysql-community-release` For CentOS7, install MySQL server from the HDP-Utils repository.

3. Start the instance.

   `/etc/init.d/mysqld 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. Log in to MySQL as the root user:

   `mysql -u root -proot`

   where "root" is the root user password.

7. Log in as the root user and 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>
```

8. Use the exit command to exit MySQL.

9. 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.

10. Install the MySQL connector JAR file.

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

1.1.5.3.2. SUSE Linux Enterprise Server (SLES)

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, enter:

   ```
   zypper install mysql-server
   ```

3. Start the instance.

```
/etc/init.d/mysqld 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. Log in to MySQL as the root user:

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

7. Log in as the root user, create 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>
```

8. Use the exit command to exit MySQL.

9. 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.

10. Install the MySQL connector JAR file.

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

1.1.5.4. Configuring Oracle as the Metastore Database

You can select Oracle as the metastore database. For instructions on how to install the databases, see your third-party documentation. To configure Oracle as the Hive Metastore, install HDP and Hive, then follow the instructions in "Set up Oracle DB for use with Hive Metastore" in this guide.

1.2. 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 is supported by HDP and any issues detected on these platforms are reproducible on the same supported operating system installed elsewhere.

See Meet Minimum System Requirements for the list of supported operating systems for HDP.

1.3. Configure the Remote Repositories

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

Note

If your cluster does not have access to the Internet, or you are creating a large cluster and you want to conserve bandwidth, you can instead provide a local copy of the HDP repository that your hosts can access. For more information, see Deploying HDP in Production Data Centers with Firewalls in the HDP Reference Guide.

- 6.x line of RHEL/CentOS/Oracle Linux

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.3.0.
0/hdp.repo
```

- 7.x line of RHEL/CentOS/Oracle Linux

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.3.0.
0/hdp.repo
```

- SLES SP3/SP4

```
wget -nv http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/updates/2.3.
0.0/hdp.repo
```

1.4. Decide on Deployment Type

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

1.5. Collect Information

To deploy your HDP, you need the following information:

- The fully qualified domain name (FQDN) for each host in your system, and the components you want to set up on each host. You can use `hostname -f` to check for the FQDN.
- If you install Hive/HCatalog or Oozie, you need the hostname, database name, username, and password for the metastore instance.

Note

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

1.6. Prepare the Environment

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

- Enable NTP on the Cluster [13]
- Disable SELinux [15]
- Disable IPTables [15]

1.6.1. Enable NTP on the Cluster

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

1. Configure NTP clients. Execute the following command on all nodes in your cluster:

   - For RHEL/CentOS/Oracle Linux 6:

     ```
     yum install ntp
     ```

   - For RHEL/CentOS/Oracle Linux 7:

   a. Configure the NTP clients:

      ```
      yum install ntp
      ```

   b. Enable the service:

      ```
      systemctl enable ntpd
      ```

   c. Start NTPD:

      ```
      systemctl start ntpd
      ```

   - For SLES:

     ```
     zypper install ntp
     ```

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

   - For RHEL/CentOS/Oracle Linux:

     ```
     chkconfig ntpd on
     ```

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

   - For RHEL/CentOS/Oracle Linux:

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

   - For SLES:

     ```
     /etc/init.d/ntp start
     ```

4. If you want to use the existing NTP server in your environment, complete the following steps:

   a. 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. For example on RHEL hosts you would use:

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

   b. Save and restart iptables. Execute the following command on all the nodes in your cluster:

      ```
      # service iptables save

      # service iptables restart
      ```

c. Finally, configure clients to use the local NTP server. Edit the `/etc/ntp.conf` file and add the following line:

```
server $LOCAL_SERVER_IP OR HOSTNAME
```

1.6.2. Disable SELinux

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

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

   ```
   getenforce
   ```

   If the command returns "disabled" or "permissive" as the response, no further actions are required. If the result is enabled, proceed to Step 2.

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

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

     ```
     setenforce 0
     ```

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

1.6.3. Disable IPTables

Certain ports must be open and available during installation. The easiest way to do this is to temporarily disable iptables. 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. See "Configuring Ports" in the *HDP Reference Guide* for more information.

- On all RHEL/CentOS 6 host machines, execute the following commands to disable iptables:

  ```
  chkconfig iptables off
  ```

  ```
  service iptables stop
  ```

  Restart iptables after your setup is complete.

- On RHEL/CENTOS 7 host machines, execute the following commands to disable firewalld:

  ```
  systemctl stop firewalld
  ```

  ```
  systemctl mask firewalld
  ```

  Restart firewalld after your setup is complete.

Important

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

1.7. Download Companion Files

We have provided a set of companion files, including script files and configuration files, that you can download. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

To download and extract the files:

```
wget http://public-repo-1.hortonworks.com/HDP/tools/2.3.0.0/
hdp_manual_install_rpm_helper_files-2.3.0.0.2557.tar.gz
tar zxvf hdp_manual_install_rpm_helper_files-2.3.0.0.2557.tar.gz
```

1.8. 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 to set 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*.

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

Hadoop Service	Parameter	Definition
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="TODO-LIST-OF-NAMENODE-DIRS";

# 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="TODO-LIST-OF-DATA-DIRS";

# 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="TODO-LIST-OF-SECONDARY-NAMENODE-DIRS";



# 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="TODO-LIST-OF-YARN-LOCAL-DIRS";

# 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="TODO-LIST-OF-YARN-LOCAL-LOG-DIRS";

# 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/mapred";

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

```
#
# 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="TODO-ZOOKEEPER-DATA-DIR";

# 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"

#
# Hadoop Service - Accumulo
#
ACCUMULO_CONF_DIR="/etc/accumulo/conf";

ACCUMULO_LOG_DIR="/var/log/accumulo"
```

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 create in Create System Users and Groups. The scripts.zip file you downloaded includes a script, usersAndGroups.sh, for setting user and group environment parameters.

Table 1.3. Define Users and Groups for Systems

Parameter	Definition
HDFS_USER	User that owns the HDFS services. For example, hdfs.
YARN_USER	User that owns the YARN services. For example, yarn.
ZOOKEEPER_USER	User that owns the ZooKeeper services. For example, zookeeper.
HIVE_USER	User that owns the Hive services. For example, hive.
WEBHCAT_USER	User that owns the WebHCat services. For example, hcat.

Parameter	Definition
HBASE_USER	User that owns the HBase services. For example, hbase.
FALCON_USER	User that owns the Falcon services. For example, falcon.
SQOOP_USER	User owning the Sqoop services. For example, sqoop.
KAFKA_USER	User owning the Kafka services. For example, kafka.
OOZIE_USER	User owning the Oozie services. For example oozie.
STORM_USER	User owning the Storm Services. For example, storm.
HADOOP_GROUP	A common group shared by services. For example, hadoop.
ACCUMULO_USER	User that owns the Accumulo services. For example, accumulo.
KNOX_USER	User that owns the Knox Gateway services. For example, knox.
NAGIOS_USER	User that owns the Nagios services. For example, nagios.

1.9. [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. Add the user to the group.

   ```
   useradd -G <groupname> <username>
   ```

2. Create the username directory.

   ```
   su hdfs
   hdfs dfs -mkdir /user/<username>
   ```

3. Give that account ownership over its directory.

   ```
   su hdfs
   hdfs dfs -chown <username>:<groupname> /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
HBase	hbase	hadoop
Falcon	falcon	hadoop

Hadoop Service	User	Group
Sqoop	sqoop	hadoop
ZooKeeper	zookeeper	hadoop
Oozie	oozie	hadoop
Knox Gateway	knox	hadoop
Nagios	nagios	nagios

1.10. Determine HDP Memory Configuration Settings

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

- Running the Yarn Utility Script [23]
- Manually Calculating YARN and MapReduce Memory Configuration Settings [24]

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.

1.10.1. Running the Yarn Utility Script

This section describes how to use the hdp-configuration-utils.py 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. See Download Companion Files.

To run the hdp-configuration-utils.py script, execute the following command from the folder containing the script `hdp-configuration-utils.py options` where options are as follows:

Table 1.5. hdp-configuration-utils.py 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

Requires python26 to run.

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

Example

Running the following command from the hdp_manual_install_rpm_helper_files-2.3.2.0.2950 directory:

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

Returns:

```
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.launch.cmd-opts =-Xmx4096m
hive.tez.container.size=6144
hive.tez.java.opts=-Xmx4096m
```

1.10.2. Manually Calculating 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 running in the cluster such as MapReduce. 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 such as memory and CPU.

In a Hadoop cluster, it is vital to balance the use 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.

Table 1.6. 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 24 GB	2 GB 4 GB	2 GB 4 GB
48 GB	6 GB	8 GB
64 GB	8 GB	8 GB
72 GB	8 GB	8 GB
96 GB 128 GB	12 GB 24 GB	16 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 DISKS is the value for dfs.data.dirs (number of data disks) per machine.

And 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:

Table 1.7. 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.

Table 1.8. YARN and MapReduce Configuration Setting Value Calculations

Configuration File	Configuration Setting	Value Calculation
yarn-site.xml	yarn.nodemanager.resource.memory-mb	= containers * RAM-per-container

Configuration File	Configuration Setting	Value Calculation
yarn-site.xml	yarn.scheduler.minimum-allocation-mb	= RAM-per-container
yarn-site.xml	yarn.scheduler.maximum-allocation-mb	= containers * RAM-per-container
mapred-site.xml	mapreduce.map.memory.mb	= RAM-per-container
mapred-site.xml	mapreduce.reduce.memory.mb	= 2 * RAM-per-container
mapred-site.xml	mapreduce.map.java.opts	= 0.8 * RAM-per-container
mapred-site.xml	mapreduce.reduce.java.opts	= 0.8 * 2 * RAM-per-container
mapred-site.xml	yarn.app.mapreduce.am.resource.mb	= 2 * RAM-per-container
mapred-site.xml	yarn.app.mapreduce.am.command-opts	= 0.8 * 2 * RAM-per-container

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

Table 1.9. Example Value Calculations

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

Table 1.10. Example Value Calculations

Configuration	Value Calculation
yarn.nodemanager.resource.memory-mb	= 17 * 2 = 34*1024 MB

Configuration	Value Calculation
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:

- Updating values for 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.
- If your installation has a large amount of 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.
- 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).

1.11. Configuring NameNode Heap Size

NameNode heap size depends on many factors such as the number of files, the number of blocks, and the load on the system. The following table provides recommendations for NameNode heap size configuration. These settings should work for typical Hadoop clusters where the number of blocks is very close to the number of files (generally the average ratio of number of blocks per file in a system is 1.1 to 1.2). Some clusters might require further tweaking of the following settings. Also, it is generally better to set the total Java heap to a higher value.

Table 1.11. NameNode Heap Size Settings

Number of Files in Millions	Total Java Heap (Xmx and Xms)	Young Generation Size (-XX:NewSize -XX:MaxNewSize)
< 1 million files	1024m	128m
1-5 million files	3072m	512m
5-10	5376m	768m
10-20	9984m	1280m
20-30	14848m	2048m
30-40	19456m	2560m
40-50	24320m	3072m
50-70	33536m	4352m

Number of Files in Millions	Total Java Heap (Xmx and Xms)	Young Generation Size (-XX:NewSize -XX:MaxNewSize)
70-100	47872m	6144m
100-125	59648m	7680m
125-150	71424m	8960m
150-200	94976m	8960m

You should also set -XX:PermSize to 128m and -XX:MaxPermSize to 256m.

The following are the recommended settings for HADOOP_NAMENODE_OPTS in the hadoop-env.sh file (replace the ##### placeholder for -XX:NewSize, -XX:MaxNewSize, -Xms, and -Xmx with the recommended values from the table):

```
-server -XX:ParallelGCThreads=8 -XX:+UseConcMarkSweepGC -XX:ErrorFile=/var/
log/hadoop/$USER/hs_err_pid%p.log -XX:NewSize=##### -XX:MaxNewSize=##### -
Xms##### -Xmx##### -XX:PermSize=128m -XX:MaxPermSize=256m -Xloggc:/var/log/
hadoop/$USER/gc.log-`date +'%Y%m%d%H%M'` -verbose:gc -XX:+PrintGCDetails -XX:
+PrintGCTimeStamps -XX:+PrintGCDateStamps -Dhadoop.security.logger=INFO,DRFAS
 -Dhdfs.audit.logger=INFO,DRFAAUDIT ${HADOOP_NAMENODE_OPTS}
```

If the cluster uses a Secondary NameNode, you should also set HADOOP_SECONDARYNAMENODE_OPTS to HADOOP_NAMENODE_OPTS in the hadoop-env.sh file:

```
HADOOP_SECONDARYNAMENODE_OPTS=$HADOOP_NAMENODE_OPTS
```

Another useful HADOOP_NAMENODE_OPTS setting is -XX:+HeapDumpOnOutOfMemoryError. This option specifies that a heap dump should be executed when an out of memory error occurs. You should also use -XX:HeapDumpPath to specify the location for the heap dump file. For example:

```
-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=./etc/heapdump.hprof
```

1.12. Allocate Adequate Log Space for HDP

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

1.13. Download the HDP Maven Artifacts

Hortonworks Release engineering team hosts all the released HDP maven artifacts at http://repo.hortonworks.com/content/repositories/releases/.

Other than the release artificats, there are certain third party artifacts that are necessary for building the HDP stack which are hosted in the Hortonworks nexus repository as well at:

http://repo.hortonworks.com/content/repositories/jetty-hadoop/

and

http://repo.hortonworks.com/content/repositories/re-hosted/

If a developer wants to develop an application against the HDP stack, and they also have a Maven repository manager in-house, then they could proxy these three repositories and continue referring to his/her internal maven groups repo.

If the developer does not have access to his in-house maven repo, they can directly use the Hortonworks public groups repo at: http://repo.hortonworks.com/content/groups/public/and continue developing applications.

2. Installing Apache ZooKeeper

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

Note

HDFS and YARN depend on ZooKeeper, so install ZooKeeper first.

2.1. Install the ZooKeeper Package

Note

In a production environment, Hortonworks recommends installing ZooKeeper server on three (or a higher odd number) nodes to ensure that ZooKeeper service will be available.

On all nodes of the cluster that you have identified as ZooKeeper servers, type:

- For RHEL/CentOS/Oracle Linux

  ```
  yum install zookeeper-server
  ```

- For SLES

  ```
  zypper install zookeeper
  ```

2.2. Securing ZooKeeper with Kerberos (optional)

Note

Before starting the following steps, refer to Setting up Security for Manual Installs.

(Optional) To secure ZooKeeper with Kerberos, perform the following steps on the host that runs KDC (Kerberos Key Distribution Center):

1. Start the kadmin.local utility:

   ```
   /usr/sbin/kadmin.local
   ```

2. Create a principal for ZooKeeper:

   ```
   sudo kadmin.local -q 'addprinc zookeeper/
   <ZOOKEEPER_HOSTNAME>@STORM.EXAMPLE.COM'
   ```

3. Create a keytab for ZooKeeper:

   ```
   sudo kadmin.local -q "ktadd -k /tmp/zk.keytab zookeeper/
   <ZOOKEEPER_HOSTNAME>@STORM.EXAMPLE.COM"
   ```

4. Copy the keytab to all ZooKeeper nodes in the cluster.

   Note

   Verify that only the ZooKeeper and Storm operating system users can access the ZooKeeper keytab.

5. Administrators must add the following properties to the zoo.cfg configuration file located at `/etc/zookeeper/conf`:

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

2.3. 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.

Hortonworks provides a set of configuration files that represent a working ZooKeeper configuration. (See Download Companion Files.) You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your ZooKeeper environment, complete the following steps to create the appropriate directories.

1. Execute the following commands on all nodes:

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

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

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

   where:

   - $ZOOKEEPER_USER is the user owning the ZooKeeper services. For example, zookeeper.

- $ZOOKEEPER_LOG_DIR is the directory to store the ZooKeeper logs. For example, `/var/log/zookeeper`.
- $ZOOKEEPER_PID_DIR is the directory to store the ZooKeeper process ID. For example, `/var/run/zookeeper`.
- $ZOOKEEPER_DATA_DIR is the directory where ZooKeeper will store data. For example, `/grid/hadoop/zookeeper/data`.

2. Initialize the ZooKeeper data directories with the 'myid' file. Create one file per ZooKeeper server, and put the number of that server in each file:

   ```
   vi $ZOOKEEPER_DATA_DIR/myid
   ```

   - In the myid file on the first server, enter the corresponding number: **1**
   - In the myid file on the second server, enter the corresponding number: **2**
   - In the myid file on the third server, enter the corresponding number: **3**

2.4. Set Up the Configuration Files

You must be set up several configuration files for ZooKeeper. Hortonworks provides a set of configuration files that represent a working ZooKeeper configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your ZooKeeper environment, complete the following steps:

1. 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.

2. Modify the configuration files.

   In the respective temporary directories, locate the `zookeeper-env.sh` file and modify the properties based on your environment including the JDK version you downloaded.

3. Edit the `zookeeper-env.sh` file to match the Java home directory, ZooKeeper log directory, ZooKeeper PID directory in your cluster environment and the directories you set up above.

   See below for an example configuration:

   ```
   export JAVA_HOME=/usr/jdk64/jdk1.8.0_40
   export ZOOKEEPER_HOME=/usr/hdp/current/zookeeper-server
   export ZOOKEEPER_LOG_DIR=/var/log/zookeeper
   export ZOOKEEPER_PID_DIR=/var/run/zookeeper/zookeeper_server.pid
   export SERVER_JVMFLAGS=-Xmx1024m
   export JAVA=$JAVA_HOME/bin/java
   CLASSPATH=$CLASSPATH:$ZOOKEEPER_HOME/*
   ```

4. Edit the zoo.cfg file to match your cluster environment. Below is an example of a typical `zoo.cfs` file:

   ```
   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
   ```

5. Copy the configuration files.

   - On all hosts create the config directory:

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

   - Copy all the ZooKeeper configuration files to the $ZOOKEEPER_CONF_DIR directory.

   - Set appropriate permissions:

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

     Note:

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

   - $ZOOKEEPER_USER is the user owning the ZooKeeper services. For example, zookeeper.

2.5. Start ZooKeeper

To install and configure HBase and other Hadoop ecosystem components, you must start the ZooKeeper service and the ZKFC:

```
sudo -E -u zookeeper bash -c "export ZOOCFGDIR=$ZOOKEEPER_CONF_DIR ; export
 ZOOCFG=zoo.cfg;
      source $ZOOKEEPER_CONF_DIR/zookeeper-env.sh ; $ZOOKEEPER_HOME/bin/
zkServer.sh
    start"
```

For example:

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

```
su -l hdfs -c "/usr/hdp/current/hadoop-hdfs-namenode/../hadoop/sbin/hadoop-
daemon.sh start zkfc"
```

3. Installing HDFS, YARN, and MapReduce

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 [34]
2. Install the Hadoop Packages [34]
3. Install Compression Libraries [34]
4. Create Directories [35]

3.1. Set Default File and Directory Permissions

Set the default operating system file and directory permissions to 0022 (022).

Use the umask command to confirm that the permissions are set as necessary. For example, to see what the current umask setting are, enter:

```
umask
```

If you want to set a default umask for all users of the OS, edit the `/etc/profile` file, or other appropriate file for system-wide shell configuration.

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

3.2. 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
  ```

3.3. Install Compression Libraries

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

3.3.1. Install Snappy

Install Snappy on all the nodes in your cluster. At each node:

- For RHEL/CentOS/Oracle Linux:

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

- For SLES:

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

3.3.2. Install LZO

Execute the following command at all the nodes in your cluster:

- RHEL/CentOS/Oracle Linux:

  ```
  yum install lzo lzo-devel hadooplzo hadooplzo-native
  ```

- For SLES:

  ```
  zypper install lzo lzo-devel hadooplzo hadooplzo-native
  ```

3.4. Create Directories

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

Before you begin:

- If any of these directories already exist, we recommend deleting and recreating them.
- Hortonworks provides a set of configuration files that represent a working ZooKeeper configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

Use the following instructions to create appropriate directories:

1. Create the NameNode Directories [35]
2. Create the SecondaryNameNode Directories [36]
3. Create DataNode and YARN NodeManager Local Directories [36]
4. Create the Log and PID Directories [37]
5. Symlink Directories with hdp-select [39]

3.4.1. 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.

3.4.2. 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.

3.4.3. Create DataNode and YARN NodeManager Local Directories

At each DataNode, 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.

At each 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 container log 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.

At each 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/logs`.
- $YARN_USER is the user owning the YARN services. For example, yarn.
- $HADOOP_GROUP is a common group shared by services. For example, hadoop.

3.4.4. Create the Log and PID Directories

Each ZooKeeper service requires a log and PID directory. In this section, you will create directories for each service. If you choose to use the companion file scripts, these environment variables will already be defined and you can copy and paste the examples into your terminal window.

3.4.4.1. HDFS Logs

At 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.

3.4.4.2. Yarn Logs

At all nodes, execute the following commands:

```
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.

3.4.4.3. HDFS Process

At all nodes, execute the following commands:

```
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.

3.4.4.4. Yarn Process ID

At all nodes, execute the following commands:

```
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.

3.4.4.5. JobHistory Server Logs

At all nodes, execute the following commands:

```
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 $MAPRED_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.

3.4.4.6. JobHistory Server Process ID

At all nodes, execute the following commands:

```
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 $MAPRED_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.

3.4.5. Symlink Directories with hdp-select

Important

HDP 2.3 installs hdp-select automatically with the installation or upgrade of the first HDP component.

To prevent version-specific directory issues for your scripts and updates, Hortonworks provides hdp-select, a script that symlinks directories to hdp-current and modifies paths for configuration directories.

Determine the version number of the hdp-select installed package:

`yum list | grep hdp (on Cent OS6)`

`rpm –q -a | grep hdp (on Cent OS7)`

`dpkg -l | grep hdp (on Ubuntu)`

For example:

```
/usr/bin/hdp-select set all 2.3.0.0-<$version>
```

Run hdp-select set all on the NameNode and on all DataNodes. If YARN is deployed separately, also run hdp-select on the Resource Manager and all Node Managers.

```

hdp-select set all 2.3.0.0-<$version>
```

4. Setting Up the Hadoop Configuration

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

You must be set up several configuration files for HDFS and MapReduce. Hortonworks provides a set of configuration files that represent a working HDFS and MapReduce configuration. (See Download Companion Files.) You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your HDFS and MapReduce environment, complete the following steps:

1. 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.

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. For further information, see "Define Environment Parameters" in this guide.

   - Edit 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>
     ```

   - Edit 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>

<property>
     <name>dfs.namenode.checkpoint.edits.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>

<property>
     <name>dfs.namenode.rpc-address</name>
     <value>namenode_host_name:8020>
     <description>The RPC address that handles all clients requests.</
description.>
</property>

<property>
     <name>dfs.namenode.https-address</name>
     <value>namenode_host_name:50470>
     <description>The namenode secure http server address and port.</
description.>
</property>
```

Note

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

- Edit `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.nodemanager.recovery</name.dir>
     <value>{hadoop.tmp.dir}/yarn-nm-recovery</value>
</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>

<property>
    <name>yarn.timeline-service.webapp.address</name>
    <value><Resource_Manager_full_hostname>:8188</value>
</property>
```

- Edit `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>
```

3. On each node of the cluster, create an empty file named dfs.exclude inside $HADOOP_CONF_DIR. Append the following to `/etc/profile`:

Note

The value of HADOOP_CONF_DIR should be the same as set in the environment earlier. See Define Environment Variables.)

```
touch $HADOOP_CONF_DIR/dfs.exclude
JAVA_HOME=<java_home_path>
export JAVA_HOME
HADOOP_CONF_DIR=/etc/hadoop/conf/
export HADOOP_CONF_DIR
export PATH=$PATH:$JAVA_HOME:$HADOOP_CONF_DIR
```

4. **Optional:** Configure MapReduce to use Snappy Compression.

   To enable Snappy compression for MapReduce jobs, edit core-site.xml and mapred-site.xml.

   - 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/hdp/current/
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/hdp/current/
hadoop/lib/native/ -Djava.net.preferIPv4Stack=true</value>
     <final>true</final>
</property>
```

   - 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:** If you are using the `LinuxContainerExecutor`, you must set up `container-executor.cfg` in the `config` directory. The file must be owned by `root:root`. The settings are in the form of `key=value` with one key per line. There must entries for all keys. If you do not want to assign a value for a key, you can leave it unset in the form of `key=#`.

   The keys are defined as follows:

   - `yarn.nodemanager.linux-container-executor.group` - the configured value of `yarn.nodemanager.linux-container-executor.group`. This must match the value of `yarn.nodemanager.linux-container-executor.group` in `yarn-site.xml`.
   - `banned.users` - a comma separated list of users who cannot run `container-executor`.
   - `min.user.id` - the minimum value of user id, this is to prevent system users from running `container-executor`.
   - `allowed.system.users` - a comma separated list of allowed system users.

6. 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. Fill in the memory/CPU values that match what the documentation or helper scripts suggests for your environment.

7. Copy the configuration files.

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

     ```
     rm -rf $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`.

   - Copy all the configuration files to $HADOOP_CONF_DIR.
   - Set the 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.

5. Validating the Core Hadoop Installation

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

5.1. Format and Start HDFS

1. Modify the JAVA_HOME value in the `hadoop-env.sh` file:

```
export JAVA_HOME=/usr/java/default
```

2. Execute the following commands on the NameNode host machine:

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

3. Execute the following commands on the SecondaryNameNode:

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

4. Execute the following commands on all DataNodes:

```
su - $HDFS_USER
/usr/hdp/current/hadoop-hdfs-datanode/../hadoop/sbin/hadoop-daemon.sh --
config $HADOOP_CONF_DIR start datanode
```

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

   Where $HDFS_USER is the HDFS user, for example, `hdfs`.

5.2. Smoke Test HDFS

1. Determine 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
   hdfs dfs -mkdir -p /user/hdfs
   ```

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

   ```
   su - $HDFS_USER
   hdfs dfs -copyFromLocal /etc/passwd passwd
   hdfs dfs -ls
   ```

4. Use the Namenode web UI and the Utilities menu to browse the file system.

5.3. Configure YARN and MapReduce

After you install Hadoop, modify your configs.

1. As the HDFS user, for example 'hdfs', upload the MapReduce tarball to HDFS.

   ```
   su - $HDFS_USER
   hdfs dfs -mkdir -p /hdp/apps/<hdp_version>/mapreduce/
   hdfs dfs -put /usr/hdp/current/hadoop-client/mapreduce.tar.gz /hdp/apps/
   <hdp_version>/mapreduce/
   hdfs dfs -chown -R hdfs:hadoop /hdp
   hdfs dfs -chmod -R 555 /hdp/apps/<hdp_version>/mapreduce
   hdfs dfs -chmod 444 /hdp/apps/<hdp_version>/mapreduce/mapreduce.tar.gz
   ```

   Where $HDFS_USER is the HDFS user, for example hdfs, and <hdp_version> is the current HDP version, for example 2.3.0.0.

2. Copy mapred-site.xml from the companion files and make the following changes to mapred-site.xml:

   - Add:

     ```
     <property>
          <name>mapreduce.admin.map.child.java.opts</name>
          <value>-server -Djava.net.preferIPv4Stack=true -Dhdp.version=${hdp.
     version}</value>
          <final>true</final>
     </property>
     ```

     Note

     You do not need to modify ${hdp.version}.

   - Modify the following existing properties to include ${hdp.version}:

     ```
      <property>
          <name>mapreduce.admin.user.env</name>
          <value>LD_LIBRARY_PATH=/usr/hdp/${hdp.version}/hadoop/lib/native:/
     usr/hdp/${hdp.version}/hadoop/lib/native/Linux-amd64-64</value>
     </property>
     ```

```
<property>
     <name>mapreduce.application.framework.path</name>
     <value>/hdp/apps/${hdp.version}/mapreduce/mapreduce.tar.gz#mr-
framework</value>
</property>

<property>
     <name>mapreduce.application.classpath</name>
     <value>$PWD/mr-framework/hadoop/share/hadoop/mapreduce/*:
     $PWD/mr-framework/hadoop/share/hadoop/mapreduce/lib/*:
     $PWD/mr-framework/hadoop/share/hadoop/common/*:
     $PWD/mr-framework/hadoop/share/hadoop/common/lib/*:
     $PWD/mr-framework/hadoop/share/hadoop/yarn/*:
     $PWD/mr-framework/hadoop/share/hadoop/yarn/lib/*:
     $PWD/mr-framework/hadoop/share/hadoop/hdfs/*:
     $PWD/mr-framework/hadoop/share/hadoop/hdfs/lib/*:/usr/hdp/${hdp.
version}/hadoop/lib/hadoop-lzo-0.6.0.${hdp.version}.jar:/etc/hadoop/conf/
secure</value>
</property>
```

Note

You do not need to modify ${hdp.version}.

3. Copy yarn-site.xml from the companion files and modify:

```
<property>
     <name>yarn.application.classpath</name>
     <value>$HADOOP_CONF_DIR,/usr/hdp/${hdp.version}/hadoop-client/*,
     /usr/hdp/${hdp.version}/hadoop-client/lib/*,
     /usr/hdp/${hdp.version}/hadoop-hdfs-client/*,
     /usr/hdp/${hdp.version}/hadoop-hdfs-client/lib/*,
     /usr/hdp/${hdp.version}/hadoop-yarn-client/*,
     /usr/hdp/${hdp.version}/hadoop-yarn-client/lib/*</value>
</property>
```

4. For secure clusters, you must create and configure the container-executor.cfg configuration file:

   - Create the container-executor.cfg file in `/etc/hadoop/conf/`.
   - Insert the following properties:

   ```
   yarn.nodemanager.linux-container-executor.group=hadoop
   banned.users=hdfs,yarn,mapred
   min.user.id=1000
   ```

   - Set the file `/etc/hadoop/conf/container-executor.cfg` file permissions to only be readable by root:

   ```
   chown root:hadoop /etc/hadoop/conf/container-executor.cfg
   chmod 400 /etc/hadoop/conf/container-executor.cfg
   ```

   - Set the container-executor program so that only root or hadoop group users can execute it:

   ```
   chown root:hadoop /usr/hdp/${hdp.version}/hadoop-yarn/bin/container-
   executor
   chmod 6050 /usr/hdp/${hdp.version}/hadoop-yarn/bin/container-executor
   ```

5.4. Start YARN

Note

To install and configure the Timeline Server see Configuring the Timeline Server.

1. As $YARN_USER, run the following command from the ResourceManager server:

```
su -l yarn -c "/usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-
daemon.sh --config $HADOOP_CONF_DIR start resourcemanager"
```

2. As $YARN_User, run the following command from all NodeManager nodes:

```
su -l yarn -c "/usr/hdp/current/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh
 --config $HADOOP_CONF_DIR start nodemanager"
```

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

5.5. Start MapReduce JobHistory Server

1. Change permissions on the container-executor file.

```
chown -R root:hadoop /usr/hdp/current/hadoop-yarn*/bin/container-executor
chmod -R 6050 /usr/hdp/current/hadoop-yarn*/bin/container-executor
```

Note

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

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

```
su $HDFS_USER
hdfs dfs -mkdir -p /mr-history/tmp
hdfs dfs -mkdir -p /mr-history/done

hdfs dfs -chmod 1777 /mr-history
hdfs dfs -chmod 1777 /mr-history/tmp
hdfs dfs -chmod 1770 /mr-history/done

hdfs dfs -chown $MAPRED_USER:$MAPRED_USER_GROUP /mr-history
hdfs dfs -chown $MAPRED_USER:$MAPRED_USER_GROUP /mr-history/tmp
hdfs dfs -chown $MAPRED_USER:$MAPRED_USER_GROUP /mr-history/done

Where
$MAPRED_USER : mapred
$MAPRED_USER_GROUP: mapred or hadoop

hdfs dfs -mkdir -p /app-logs
hdfs dfs -chmod 1777 /app-logs
hdfs dfs -chown $YARN_USER:$HADOOP_GROUP /app-logs

Where
$YARN_USER : yarn
$HADOOP_GROUP: hadoop
```

3. Run the following command from the JobHistory server:

```
su -l $YARN_USER -c "/usr/hdp/current/hadoop-mapreduce-historyserver/sbin/
mr-jobhistory-daemon.sh --config $HADOOP_CONF_DIR start historyserver"
```

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

5.6. Smoke Test MapReduce

1. Browse to the ResourceManager:

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

2. Create a $CLIENT_USER in all of the nodes and add it to the **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, sort 10GB of data.

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

6. Installing Apache 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.

Note

You must install and configure ZooKeeper prior to installing HBase. See Installing ZooKeeper.

6.1. Install the HBase Package

Prerequisites

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

   ```
   yum list hbase
   ```

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

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

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

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

Installation

On hosts identified as HBase Master/RegionServers, type:

The files are located in the `configuration_files/hbase` directory in the companion files.

- For RHEL/CentOS/Oracle Linux

```
yum install hbase
```

- For SLES

```
zypper install hbase
```

6.2. Set Directories and Permissions

Create directories and configure ownership and permissions on the appropriate hosts as described below. Hortonworks provides a set of configuration files that represent a working ZooKeeper configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your ZooKeeper environment, complete the following steps to create the appropriate directories. If any of these directories already exist, we recommend deleting and recreating them.

1. 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

You must set up several configuration files for HBase and ZooKeeper. Hortonworks provides a set of configuration files that represent a working ZooKeeper configuration. (See Download Companion Files). You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your ZooKeeper environment, complete the following steps:

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

The files are located in the `configuration_files/hbase` directory in the companion files.

2. Modify the configuration files.

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

   - Review the `zoo.cfg` file and locate the ZooKeeper servers.

     ```
     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
     ```

   - Edit 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>
     ```

   - If you are using a REST server to connect to HBase secured by Kerberos:

     - You must also add the following properties to `hbase-site.xml`:

       ```
       <property>
            <name>hbase.rest.authentication.type</name>
            <value>kerberos</value>
            <description>Enter the authentication method for the REST server.</
       description>
       </property>

       <property>
            <name>hbase.rest.kerberos.principal</name>
            <value>hbase/_HOST@EXAMPLE.COM</value>
            <description>Enter the Kerberos principal for the REST server to
        use to interact with HBase.</description>
       </property>

       <property>
            <name>hbase.rest.keytab.file</name>
            <value>/etc/security/keytabs/hbase.service.keytab</value>
            <description>Enter the location of the keytab file for the REST
        server to use to interact with HBase.</description>
       </property>

       <property>
       ```

```
     <name>hbase.rest.authentication.kerberos.principal</name>
     <value>HTTP/_HOST@EXAMPLE.COM</value>
     <description>Enter the Kerberos principal for accepting SPNEGO-
authenticated REST requests.</description>
</property>

<property>
     <name>hbase.rest.authentication.kerberos.keytab</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
     <description>Enter the location of the keytab file for accepting
 SPNEGO-authenticated REST requests.</description>
</property>
```

Important

You must set the primary component part of the value for hbase.rest.authentication.kerberos.principal to **HTTP**. SPNEGO authentication **requires** that the Kerberos principal's primary component (the first element, up to the forward-slash ("/") or at-symbol ("@") to be **HTTP**.

- After adding these properties to the `hbase-site.xml` file, you must grant HBase permissions to the user specified by the value of the hbase.rest.kerberos.principal property:

  ```
  grant '<user-name>', '<permissions>', '<table>' [, '<column-family>' [,
   '<column-qualifier>']]
  ```

  For example, if user = HTTP, permissions = RWXCA, table = `sales`, and column = `1`:

  ```
  grant 'hbase', 'RWXCA', 'sales', '1'
  ```

- Ensure that the `core-site.xml` file also contains the corresponding proxy user configuration properties for the configured REST server user.

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

  <property>
      <name>hadoop.proxyuser.USER.groups</name>
      <value>*</value>
  </property>

  <property>
      <name>hadoop.proxyuser.USER.users</name>
      <value>*</value>
  </property>
  ```

  For example, if user = `hbase` and we wanted to allow the users `alice` and `bob` to be impersonated only from the REST server host `10.0.0.1`

```
<property>
    <name>hadoop.proxyuser.hbase.hosts</name>
    <value>10.0.0.1</value>
</property>

<property>
    <name>hadoop.proxyuser.hbase.users</name>
    <value>alice,bob</value>
</property>
```

- 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. Use full host names (FQDNs).

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

3. Copy the configuration files.

- On all hosts create the config directory:

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

- Copy all of the HBase configuration files to the $HBASE_CONF_DIR.

- 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.

4. Review `hbase-site.xml` and `hbase-env.sh`. In the `hbase-env.sh` file, check the Java heap size for HBase master and Region servers (Xms and Xmx settings in HBASE_MASTER_OPTS and HBASE_REGIONSERVER_OPTS). Compare the Region server heap size with the recommended HBase memory values listed in Table 1.6 in Determine HDP Memory Configuration Settings.

6.4. Validate the Installation

Use these steps to validate your installation.

1. Start HBase.

   - Execute the following command from the HBase Master node:

     ```
     su -l hbase -c "/usr/hdp/current/hbase-master/bin/hbase-
     daemon.sh start master; sleep 25"
     ```

   - Execute the following command from each HBase Region Server node:

     ```
     su -l hbase -c "/usr/hdp/current/hbase-regionserver/bin/hbase-
     daemon.sh start regionserver"
     ```

2. Smoke Test HBase.

   From a terminal window, enter:

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

   In the HBase shell, enter the following command:

   ```
   status 'detailed'
   ```

6.5. Start the HBase Thrift and REST Servers

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

Starting the HBase Thrift and REST Servers in the Foreground

Where <port> is the service’s port, and <info port> is the port for the web-ui with information about the service, use the following command to start the HBase Thrift server in the foreground:

```
hbase thrift start -p <port> --infoport <infoport>
```

Where <port> is the service’s port, and <info port> is the port for the web-ui with information about the service, use the following command to start the HBase REST server in the foreground:

```
hbase rest start -p <port> --infoport <infoport>
```

Starting the HBase Thrift Server in the Background

Where <port> is the service’s port, and <info port> is the port for the web-ui with information about the service, use the following command to start the HBase Thrift server in the background:

```
/usr/hdp/current/hbase-master/bin/hbase-daemon.sh start thrift -p <port> --
infoport <infoport>
```

Where <port> is the service’s port, and <info port> is the port for the web-ui with information about the service, use the following command to start the HBase REST server in the background:

```
/usr/hdp/current/hbase-master/bin/hbase-daemon.sh start rest -p <port> --
infoport <infoport>
```

For additional information, see the Starting and Stopping the REST Server and Thrift API and Filter Language sections of Apache HBase Reference Guide.

7. Installing Apache Phoenix

To install Apache Phoenix, complete the following instructions on all HBase Region Servers and all Master nodes.

7.1. Installing the Phoenix Package

Run the following command to install Phoenix:

- RHEL/CentOS/Oracle Linux

  ```
  yum install phoenix
  ```

- SLES

  ```
  zypper install phoenix
  ```

7.2. Configuring HBase for Phoenix

To enable global indexing and local indexing in Phoenix, complete the following steps.

1. Add the following properties to the hbase-site.xml file on all HBase nodes, the Master Server, and all Region servers.

   - Set hbase.defaults.for.version.skip to true:

     ```
     <property>
          <name>hbase.defaults.for.version.skip</name>
          <value>true</value>
     </property>
     ```

   - Set hbase.regionserver.wal.codec to enable custom write-ahead log ("WAL") edits to be written as follows:

     ```
     <property>
          <name>hbase.regionserver.wal.codec</name>
          <value>org.apache.hadoop.hbase.regionserver.wal.IndexedWALEditCodec</
     value>
     </property>
     ```

   - Set the following properties to prevent deadlocks from occurring during index maintenance for global indexes by ensuring index updates are processed with a higher priority than data updates and also to prevent deadlocks by ensuring metadata rpc calls are processed with a higher priority than data rpc calls.

```
<property>
  <name>hbase.region.server.rpc.scheduler.factory.class</name>
  <value>org.apache.hadoop.hbase.ipc.PhoenixRpcSchedulerFactory</value>
  <description>Factory to create the Phoenix RPC Scheduler that uses
 separate queues for index and metadata updates</description>
</property>

<property>
  <name>hbase.rpc.controllerfactory.class</name>
  <value>org.apache.hadoop.hbase.ipc.controller.
ServerRpcControllerFactory</value>
  <description>Factory to create the Phoenix RPC Scheduler that uses
 separate queues for index and metadata updates</description>
</property>
```

2. To enable user-defined functions, configure the following property in `hbase-site.xml` on all Hbase nodes.

```
<property>
 <name>phoenix.functions.allowUserDefinedFunctions</name>
 <value>true</value>
 <description>enable UDF functions</description>
</property>
```

3. (**Optional**) To use local indexing, set the following properties in `hbase-site.xml` on HBase Master. These properties ensure co-location of data table and local index regions:

Warning

The local indexing feature is a technical preview and considered under development. Do not use this feature in your production systems. If you have questions regarding this feature, contact Support by logging a case on our Hortonworks Support Portal.

```
<property>
     <name>hbase.master.loadbalancer.class</name>
     <value>org.apache.phoenix.hbase.index.balancer.IndexLoadBalancer</
value>
</property>

<property>
     <name>hbase.coprocessor.master.classes</name>
     <value>org.apache.phoenix.hbase.index.master.IndexMasterObserver</
value>
</property>

<property>
     <name>hbase.coprocessor.regionserver.classes</name>
     <value>org.apache.hadoop.hbase.regionserver.LocalIndexMerger</value>
</property>
```

4. Restart the HBase Master and Region Servers.

7.3. Configuring Phoenix to Run in a Secure Cluster

To configure Phoenix to run in a secure Hadoop cluster, set HBASE_CONF_PATH as follows:

```
export HBASE_CONF_PATH=HBASE_CONFIG_DIR:HADOOP_CONFIG_DIR
```

For example:

```
export HBASE_CONF_PATH=/etc/hbase/conf:/etc/hadoop/conf
```

Alternately, you can use the pre-2.2 method:

1. Link the HBase configuration file with the Phoenix libraries:

   ```
   ln -sf <HBASE_CONFIG_DIR>/hbase-site.xml /usr/hdp/current/
   phoenix-client/bin/hbase-site.xml
   ```

2. Link the Hadoop configuration file with the Phoenix libraries:

   ```
   ln -sf <HADOOP_CONFIG_DIR>/core-site.xml /usr/hdp/current/
   phoenix-client/bin/core-site.xml
   ```

   ```
   ln -sf <HADOOP_CONFIG_DIR>/hdfs-site.xml /usr/hdp/current/
   phoenix-client/bin/hdfs-site.xml
   ```

Note

When running the pssql.py and sqlline.py Phoenix scripts in secure mode, you can safely ignore the following warnings:

```
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,
ignoredjava.io.IOException: No FileSystem for scheme: hdfs
```

7.4. Validating the Phoenix Installation

Validating a Native Phoenix Installation on an Unsecured Cluster

To validate your installation, log in as the hbase user, navigate to the Phoenix home directory, and run the following smoke tests:

```
cd /usr/hdp/current/phoenix-client/bin/ ./psql.py
localhost:2181:/hbase-unsecure
/usr/hdp/current/phoenix-client/doc/examples/WEB_STAT.sql
/usr/hdp/current/phoenix-client/doc/examples/WEB_STAT.csv
/usr/hdp/current/phoenix-client/doc/examples/WEB_STAT_QUERIES.sql
```

Where `localhost` is your ZooKeeper node.

Validating a Native Phoenix Installation on a Cluster Secured with Kerberos

To validate your installation, log in as the hbase user, and perform the following actions:

1. Set the HBASE_CONF_PATH for a secured cluster:

   ```
   export HBASE_CONF_PATH=/etc/hbase/conf:/etc/hadoop/conf
   ```

   Note

   For more information, see Configuring Phoenix to Run in a Secure Cluster

2. Obtain a valid Kerberos ticket by running `kinit`. For example:

   ```
   kinit -k -t /etc/security/keytabs/hbase.headless.keytab hbase
   ```

3. Navigate to the Phoenix home directory, and run the following smoke tests:

   ```
   cd /usr/hdp/current/phoenix-client/bin/ ./psql.py
   localhost:2181:/hbase-unsecure
   /usr/hdp/current/phoenix-client/doc/examples/WEB_STAT.sql
   /usr/hdp/current/phoenix-client/doc/examples/WEB_STAT.csv
   /usr/hdp/current/phoenix-client/doc/examples/WEB_STAT_QUERIES.sql
   ```

   Where `localhost` is your ZooKeeper node and you replace `/hbase-unsecure` with your secured ZooKeeper node. Check the value of `zookeeper.znode.parent` in the `hbase-site.xml` configuration file to confirm the directory path.

Validating the JDBC Connection to External Applications

If you are running external applications, it is recommended that you test the connection to HBase using the following connection strings for the Phoenix JDBC driver:

1. Add hbase-site.xml and core-site.xml to your application or client's class path:

   ```
   set CLASSPATH=<path_to_hbase-site.xml>;<path_to_core-site.xml>
   ```

2. Depending on whether you have an unsecured cluster or a cluster secured with Kerberos, use one of the following connection strings to connect to HBase.

   - **For unsecured clusters:**

     ```
     jdbc:phoenix:<ZooKeeper_host_name>:<port_number>:<root_node>
     ```

     Where <*ZooKeeper_host_name*> can specify one host or several hosts. If you specify several ZooKeeper hosts, insert commas between host names. For example, <ZK_host1, ZK_host2, ZK_host3>.

     Example:

     ```
     jdbc:phoenix:zk_quorum:2181:zk_parent
     ```

   - **For clusters secured with Kerberos:**

     ```
     jdbc:phoenix:<ZooKeeper_host_name>:<port_number>:<secured_ZooKeeper_node>:<principal_nam
     ```

     Where <*secured_ZooKeeper_node*> is the path to the secured ZooKeeper node, and <*HBase_headless_keytab_file*> is the location of this keytab file.

Example:

```
jdbc:phoenix:zk_quorum:2181:/hbase-secure:hbase@EXAMPLE.COM:/hbase-secure/
keytab/keytab_file
```

Note

If any part of the connection string is omitted, the corresponding property value (hbase.zookeeper.quorum, hbase.zookeeper.property.clientPort, or zookeeper.znode.parent) from the hbase-site.xml configuration file is used. 2181 is the default port.

7.5. Troubleshooting Phoenix

You might 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:

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

8. Installing and Configuring Apache Tez

Apache Tez is an extensible YARN framework that can be used to build high-performance batch and interactive data processing applications. Tez dramatically improves MapReduce processing speed while maintaining its ability to scale to petabytes of data. Tez can also be used by other Hadoop ecosystem components such as Apache Hive and Apache Pig to dramatically improve query performance.

8.1. Prerequisites

Verify that your cluster is upgraded to HDP 2.3 or higher before installing Tez.

Note

Please note that the instructions for installing and configuring Tez on HDP 2.3 are different than the instructions for Tez on HDP 2.1 and on HDP 2.2.

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

8.2. Installing the Tez Package

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 from any one of the cluster client nodes to upload the Tez tarball into HDFS:

   ```
   su - $HDFS_USER
   hdfs dfs -mkdir -p /hdp/apps/<hdp_version>/tez/
   hdfs dfs -put /usr/hdp/<hdp_version>/tez/lib/tez.tar.gz /hdp/apps/
   <hdp_version>/tez/
   hdfs dfs -chown -R $HDFS_USER:$HADOOP_USER /hdp
   hdfs dfs -chmod -R 555 /hdp/apps/<hdp_version>/tez
   ```

```
hdfs dfs -chmod -R 444 /hdp/apps/<hdp_version>/tez/tez.tar.gz
```

Where:

$HDFS_USER is the user that owns the HDFS service. For example, hdfs. <hdp_version> is the current HDP version, such as 2.3.0.0.

3. Execute the following command to verify that the files were copied in Step 2:

```
su - $HDFS_USER
hdfs dfs -ls /hdp/apps/<hdp_version>/tez
```

This command returns a message similar to the following:

```
Found 1 items
-r--r--r-- 3 hdfs hadoop 36518223 2015-02-12 15:35 /hdp/apps/2.3.0.0-2800/
tez/tez.tar.gz
```

8.3. Configuring Tez

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. In the `tez-site.xml` file, configure the tez.lib.uris property with the HDFS path containing the Tez tarball file.

```
...
<property>
     <name>tez.lib.uris</name>
     <value>/hdp/apps/<hdp_version>/tez/tez.tar.gz</value>
</property>
...
```

Where <hdp_version> is the current HDP version, such as 2.3.0.0.

Table 8.1. Tez Configuration Parameters

Configuration Parameter	Description	Default Value
tez.am.acls.enabled	Enables or disables access control list checks on Application Master (AM) and history data.	true
tez.am.am-rm.heartbeat.interval-ms.max	The maximum heartbeat interval between the AM and RM in milliseconds.	250
tez.am.client.am.port-range	Range of ports that the AM can use when binding for client connections. Leave this blank to use all possible ports.	No default setting. The format is a number range. For example, `10000-19999`
tez.am.container.idle.release-timeout-max.millis	The maximum amount of time to hold on to a container if no task can be assigned to it immediately. Only active when reuse is enabled.	20000
tez.am.container.idle.release-timeout-min.millis	The minimum amount of time to hold on to a container that is idle. Only active when reuse is enabled.	10000

Configuration Parameter	Description	Default Value
tez.am.container.reuse.enabled	Configuration that specifies whether a container should be reused.	true
tez.am.container.reuse.locality.delay-allocation-millis	The amount of time to wait before assigning a container to the next level of locality. NODE -> RACK -> NON_LOCAL	250
tez.am.container.reuse.non-local-fallback.enabled	Specifies whether to reuse containers for non-local tasks. Active only if reuse is enabled.	false
tez.am.container.reuse.rack-fallback.enabled	Specifies whether to reuse containers for rack local tasks. Active only if reuse is enabled.	true
tez.am.launch.cluster-default.cmd-opts	**Note:** This property should only be set by administrators – it should not be used by non-administrative users. Cluster default Java options for the Tez AppMaster process. These will be prepended to the properties specified with tez.am.launch.cmd-opts.	-server -Djava.net.preferIPv4Stack=true -Dhdp.version=${hdp.version}
tez.am.launch.cmd-opts	Command line options that are provided during the launch of the Tez `AppMaster` process. Do not set any Xmx or Xms in these launch options so that Tez can determine them automatically.	-XX:+PrintGCDetails -verbose:gc -XX:+PrintGCTimeStamps -XX:+UseNUMA -XX:+UseParallelGC
tez.am.launch.env	Environment settings for the Tez `AppMaster` process.	LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$HADOOP_COMMON_HOME/lib/native/
tez.am.log.level	Root logging level passed to the Tez Application Master. Simple configuration: Set the log level for all loggers. For example, set to `INFO`. This sets the log level to INFO for all loggers. **Advanced configuration:** Set the log level for all classes, along with a different level for some classes. For example, set to `DEBUG;org.apache.hadoop.ipc=INFO;org.apache.hadoop.security=INFO` This sets the log level for all loggers to DEBUG, except for `org.apache.hadoop. ipc` and `org.apache.hadoop.security`, which are set to INFO. **Note:**The global log level must always be the first parameter. For example: `DEBUG;org.apache.hadoop. ipc=INFO;org.apache. hadoop.security=INFO` is valid. `org.apache.hadoop.ipc=INFO;org.apache.hadoop. security=INFO` is *not* valid.	INFO
tez.am.max.app.attempts	Specifies the total number of times that the app master is re-run in case recovery is triggered.	2

Configuration Parameter	Description	Default Value
tez.am.maxtaskfailures.per.node	The maximum number of allowed task attempt failures on a node before it gets marked as blacklisted.	10
tez.am.modify-acls	Enables specified users or groups to modify operations on the AM such as submitting DAGs, pre-warming the session, killing DAGs, or shutting down the session. Format: comma-separated list of users, followed by a whitespace, and then a comma-separated list of groups. For example, `"lhale,msmith administrators,users"`	No default setting
tez.am.resource.cpu.vcores	The number of virtual cores to be used by the `AppMaster` process. Set this to > 1 if the RM Scheduler is configured to support virtual cores.	1
tez.am.resource.memory.mb	The amount of memory to be used by the AppMaster. Used only if the value is not specified explicitly by the DAG definition.	1536
tez.am.session.min.held-containers	The minimum number of containers that will be held in session mode. Not active in non-session mode. Enables an idle session that is not running a DAG to hold on to a minimum number of containers to provide fast response times for the next DAG.	0
tez.am.task.max.failed.attempts	The maximum number that can fail for a particular task before the task fails. This does not count killed attempts. A task failure results in a DAG failure. Must be an integer.	4
tez.am.view-acls	AM view ACLs. This setting enables the specified users or groups to view the status of the AM and all DAGs that run within the AM. Format: a comma-separated list of users, a whitespace, and then a comma-separated list of groups. For example, `"lhale,msmith administrators,users"`	No default value
tez.cluster.additional.classpath.prefix	Specify additional classpath information to be used for Tez AM and all containers. This will be prepended to the classpath before all framework specific components have been specified.	/usr/hdp/${hdp.version}/hadoop/lib/hadoop-lzo-0.6.0.${hdp.version}.jar:/etc/hadoop/conf/secure
tez.container.max.java.heap.fraction	A double value. Tez automatically determines the Xmx for the Java virtual machines that are used to run Tez tasks and Application Masters. This is enabled if the Xmx or Xms values have not been specified in the launch command options. Automatic Xmx calculation is preferred because Tez can determine the best value based on the actual allocation of memory to tasks in the cluster. The	0.8

Configuration Parameter	Description	Default Value
	value should be greater than 0 and less than 1.	
tez.counters.max	The number of allowed counters for the executing DAG.	2000
tez.counters.max.groups	The number of allowed counter groups for the executing DAG.	1000
tez.generate.debug.artifacts	Generates debug artifacts such as a text representation of the submitted DAG plan.	false
tez.grouping.max-size	Upper size limit (in bytes) of a grouped split, to avoid generating an excessively large split. Replaces tez.am.grouping.max-size	1073741824 (1 GB)
tez.grouping.min-size	Lower size limit (in bytes) of a grouped split, to avoid generating too many splits.	52428800 (50 MB)
tez.grouping.split-waves	The multiplier for available queue capacity when determining number of tasks for a Vertex. When set to its default value of 1.7 with 100% queue available implies generating a number of tasks roughly equal to 170% of the available containers on the queue.	1.7
tez.history.logging.service.class	The class to be used for logging history data. Set to org.apache.tez.dag.history.logging. ats.ATSHistoryLoggingService to log to ATS. Set to org.apache.tez. dag.history.logging.impl. SimpleHistoryLoggingService to log to the filesystem specified by ${fs.defaultFS}.	org.apache.tez.dag.history.logging. ats.ATSHistoryLoggingService
tez.lib.uris	Comma-delimited list of the location of the Tez libraries which will be localized for DAGs. Specifying a single .tar.gz or .tgz assumes that a compressed version of the tez libs is being used. This is uncompressed into a tezlibs directory when running containers, and tezlibs/;tezlibs/lib/ are added to the classpath (after . and .*). If multiple files are specified - files are localized as regular files, contents of directories are localized as regular files (non-recursive).	/hdp/apps/<hdp_version>/tez/ tez.tar.gz
tez.queue.name	This property should not be set in `tez-site.xml`. Instead, it can be provided on the command line when you are launching a job to determine which YARN queue to submit a job to.	No default setting
tez.runtime.compress	Specifies whether intermediate data should be compressed or not.	true
tez.runtime.compress.codec	The codec to be used if compressing intermediate data. Only applicable if tez.runtime.compress is enabled.	org.apache.hadoop.io.compress. SnappyCodec
tez.runtime.io.sort.factor	The number of streams to merge at once while sorting files. This determines the number of open file handles.	10

Configuration Parameter	Description	Default Value
tez.runtime.io.sort.mb	The size of the sort buffer when output is sorted.	512
tez.runtime.sorter.class	Which sorter implementation to use. Valid values: • `LEGACY` • `PIPELINED` The legacy sorter implementation is based on the Hadoop MapReduce shuffle implementation. It is restricted to 2GB memory limits. Pipeline sorter is a more efficient sorter that supports > 2GB sort buffers.	PIPELINED
tez.runtime.sort.spill.percent	The soft limit in the serialization buffer. Once this limit is reached, a thread begins to spill the contents to disk in the background. **Note:**Collection will not block if this threshold is exceeded while a spill is already in progress, so spills can be larger than this threshold when it is set to less than .5	0.8
tez.runtime.unordered.output.buffer.size-mb	The size of the buffer when output is not sorted.	100
tez.session.am.dag.submit.timeout.secs	Time (in seconds) for which the Tez AM should wait for a DAG to be submitted before shutting down.	300
tez.session.client.timeout.secs	Time (in seconds) to wait for AM to come up when trying to submit a DAG from the client.	-1
tez.shuffle-vertex-manager.max-src-fraction	In case of a ScatterGather connection, once this fraction of source tasks have completed, all tasks on the current vertex can be scheduled. Number of tasks ready for scheduling on the current vertex scales linearly between min-fraction and max-fraction.	0.4
tez.shuffle-vertex-manager.min-src-fraction	In case of a ScatterGather connection, the fraction of source tasks which should complete before tasks for the current vertex are scheduled.	0.2
tez.staging-dir	The staging directory used while submitting DAGs.	/tmp/${user.name}/staging
tez.task.am.heartbeat.counter.interval-ms.max	Time interval at which task counters are sent to the AM.	4000
tez.task.generate.counters.per.io	Sets whether to generate counters per IO or not. Enabling this will rename CounterGroups/ CounterNames, making them unique per vertex edge instead of unique per vertex.	true
tez.task.get-task.sleep.interval-ms.max	Maximum amount of time, in seconds, to wait before a task asks an AM for another task.	200

Configuration Parameter	Description	Default Value
tez.task.launch.cluster-default.cmd-opts	**Note:** This property should only be set by administrators – it should not be used by non-administrative users. Cluster default Java options for tasks. These will be prepended to the properties specified with tez.task.launch.cmd-opts	-server -Djava.net.preferIPv4Stack=true -Dhdp.version=${hdp.version}
tez.task.launch.cmd-opts	Java options for tasks. The Xmx value is derived based on tez.task.resource.memory.mb and is 80% of this value by default. Used only if the value is not specified explicitly by the DAG definition.	-XX:+PrintGCDetails -verbose:gc -XX:+PrintGCTimeStamps -XX:+UseNUMA -XX:+UseParallelGC
tez.task.launch.env	Additional execution environment entries for Tez. This is not an additive property. You must preserve the original value if you want to have access to native libraries. Used only if the value is not specified explicitly by the DAG definition.	LD_LIBRARY_PATH=/usr/hdp/${hdp.version}/hadoop/lib/native:/usr/hdp/${hdp.version}/hadoop/lib/native/Linux-amd64-64/
tez.task.log.level	Root logging level that is passed to the Tez tasks. Simple configuration: Set the log level for all loggers. For example, set to `INFO`. This sets the log level to INFO for all loggers. **Advanced configuration:** Set the log level for all classes, along with a different level for some classes. For example, set to `DEBUG;org.apache.hadoop.ipc=INFO;org.apache.hadoop.security=INFO` This sets the log level for all loggers to DEBUG, except for `org.apache.hadoop.ipc` and `org.apache.hadoop.security`, which are set to INFO. **Note:**The global log level must always be the first parameter. For example: `DEBUG;org.apache.hadoop.ipc=INFO;org.apache.hadoop.security=INFO` is valid. `org.apache.hadoop.ipc=INFO;org.apache.hadoop.security=INFO` is *not* valid.	INFO
tez.task.max-events-per-heartbeat	Maximum number of events to fetch from the AM by the tasks in a single heartbeat.	500
tez.task.resource.cpu.vcores	The number of virtual cores to be used by the Tez tasks. Set this to > 1 if RM Scheduler is configured to support virtual cores.	1
tez.task.resource.memory.mb	The amount of memory to be used by launched tasks. Used only if the value is not specified explicitly by the DAG definition.	1024

Configuration Parameter	Description	Default Value
tez.use.cluster.hadoop-libs	Specifies whether Tez will use the cluster Hadoop libraries. This property should not be set in `tez-site.xml`, or if it is set, the value should be false.	false

Note

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

To monitor the progress of a Tez job or to analyze the history of a Tez job, set up the Tez View in Ambari. For information about setting up the Tez view, see Configuring Your Cluster for Tez View in the HDP Ambari Views Guide.

8.4. Creating a New Tez View Instance

To create a new Tez View instance, refer to Creating or Editing the Tez View Instance. Tez Views are part of the Ambari Views Framework, which allows developers to create UI components that "plug into" the Ambari Web interface. These views can also be used in manual configurations with components such as Tex.

8.5. Validating 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. Create a `/tmp/input/` directory in HDFS and copy the test.txt file into that directory:

```
hdfs dfs -mkdir -p /tmp/input/
hdfs dfs -put test.txt /tmp/input/
```

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

```
hadoop jar /usr/hdp/current/tez-client/tez-examples-*.jar orderedwordcount /
tmp/input/test.txt /tmp/out
```

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

```
hdfs dfs -cat '/tmp/out/*'
```

This should return:

```
foo 3
bar 2
```

8.6. Troubleshooting

To troubleshoot your Tez installation and configuration, refer first to Using the Tez View. You can also 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]
```

The application ID is listed at the end of the output for a running application, as shown below in the OrderedWordCount application output:

```

14/02/26 14:45:33 INFO examples.OrderedWordCount: DAG 1 completed. FinalState=
 SUCCEEDED14/02/26
14:45:33 INFO client.TezSession: Shutting down Tez Session, sessionName=
OrderedWordCountSession, applicationId=application_1393449467938_0001
```

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:

9.1. Installing the Hive-HCatalog Package

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-hcatalog
```

- For SLES:

```
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, Derby is not recommended for production use. Use MySQL, Oracle, SQL Server, or Postgres for production use.

   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.

   Important

   If you are using MySQL, you must use `mysql-connector-java-5.1.35.zip` or later JDBC driver.

   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/hdp/current/hive-client/lib/` directory on your Hive host machine.

   c. Verify that the .jar file has at least the minimum set of permissions required.

9.2. Setting 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.

Hortonworks provides a set of configuration files that represent a working Hive/HCatalog configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your Hive/HCatalog environment, complete the following steps to create the appropriate directories. If any of these directories already exist, we recommend deleting and recreating them.

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

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

9.3. Setting Up the Hive/HCatalog Configuration Files

Note

When using HiveServer2 in HTTP mode, you must configure the mapping from Kerberos Principals to short names in the “hadoop.security.auth_to_local" property setting in the core-site.xml file.

Use the following instructions to set up the Hive/HCatalog configuration files. Hortonworks provides a set of configuration files that represent a working Hive/HCatalog configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your Hive/HCatalog environment, complete the following steps:

1. Extract the configuration files to a temporary directory.

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

   Edit the connection properties for your Hive metastore database in `hive-site.xml` to match your own cluster environment.

Warning

To prevent memory leaks in unsecure mode, disable file system caches by setting the following parameters to true in hive-site.xml:

- `fs.hdfs.impl.disable.cache`
- `fs.file.impl.disable.cache`

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

```
<property>
     <name>hive.metastore.pre-event.listeners</name>
     <value>org.apache.hadoop.hive.ql.security.authorization.
AuthorizationPreEventListener</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.

4. 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=""
```

5. **(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>
```

Important

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.

9.3.1. HDP-Utility script

You can also use the HDP utility script to fine-tune memory configuration settings based on node hardware specifications. For information on where to get the HDP-Utility script, see Determine HDP Memory Confguration Settings

Copy the configuration files.

- On all Hive hosts create the Hive configuration directory:

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

- Copy all the configuration files to the $HIVE_CONF_DIR directory.
- 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.

9.3.2. 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.xmlconnection properities for your Hive metastore database, the only remaining task would be to adjust hive.tez.container.size and hive.tez.java.opts values as described in the following section. You can also use the HDP utility script described earlier in this guide to calculate these Tez memory configuration settings.

9.3.2.1. 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 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.	-1 (not specified) If this is not specified, the memory settings from the MapReduce configurations (mapreduce.map.memory.mb) will be used by default for map tasks.
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.
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 be 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 might improve performance of HiveServer2 by increasing this number.
hive.server2.tez.initialize.default. sessions	Enables a user to use HiveServer2 without enabling Tez for HiveServer2. Users might potentially 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

9.3.2.2. 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.

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

9.3.2.3. 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.

9.4. Setting Up the Database for the Hive Metastore

Use the following steps to set up the database for your Hive Metastore. This step must be performed when you start the metastore for the first time.

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/hdp/current/hive`.

2. Turn off autocreation of schemas. 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 -c "nohup /grid/0/hdp/current/hive-metastore/
bin/hive --service metastore>/var/log/hive/hive.out 2>/var/log/
hive/hive.log &"
```

Note

You might receive the following error after running the `su - hive` command:

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

If you get this error, you might 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/nologin63
```

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.

   • Open Hive command line shell by entering the following in a terminal window:

   ```
   hive
   ```

   • Run sample commands:

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

5. Start HiveServer2:

-
  ```
  su - $HIVE_USER
  nohup /usr/hdp/current/hive-server2/bin/hiveserver2 >/var/log/hive/
  hiveserver2.out 2> /var/log/hive/hiveserver2.log &
  ```

6. Smoke test HiveServer2:

   - Open Beeline command line shell to interact with HiveServer2:

     ```
     /usr/hdp/current/hive/bin/beeline
     ```

   - Establish connection to server:

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

   - Run sample commands:

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

9.5. Setting up RDBMS for use with Hive Metastore

Hive supports multiple databases. This section uses Oracle as an example. To use OracleDB as the Hive Metastore database, you must have already installed HDP and Hive.

Important

When Hive is configured to use an Oracle database and transactions are enabled in Hive, queries might fail with the error `org.apache.hadoop.hive.ql.lockmgr.LockException: No record of lock could be found, might have timed out`. This can be caused by a bug in the BoneCP connection pooling library. In this case, Hortonworks recommends that you set the `datanucleus.connectionPoolingType` property to `dbcp` so the DBCP component is used.

To set up Oracle for use with Hive:

1. On the Hive Metastore host, install the appropriate JDBC .jar file.

   - Download the Oracle JDBC (OJDBC) driver.

   - Select "Oracle Database 11g Release 2 - ojdbc6.jar"

   - Copy the .jar file to the Java share directory:

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

     #### Note

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

2. Create a user for Hive and grant it permissions using SQL*Plus, 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.

9.6. Creating Directories on HDFS

1. Create the Hive user home directory on HDFS.

   Login as $HDFS_USER and run the following command:

   ```
   hdfs dfs -mkdir -p /user/$HIVE_USER
   hdfs dfs -chown $HIVE_USER:$HDFS_USER /user/$HIVE_USER
   ```

2. Create the warehouse directory on HDFS.

   Login as $HDFS_USER and run the following command:

   ```
   hdfs dfs -mkdir -p /apps/hive/warehousehadoop
   hdfs dfs -chown -R $HIVE_USER:$HDFS_USER /apps/hive
   hdfs dfs -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 and run the following command:

   ```
   hdfs dfs -mkdir -p /tmp/hive
   hdfs dfs -chown -R $HIVE_USER:$HDFS_USER /tmp/hive
   hdfs dfs -chmod -R 777 /tmp/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`.

9.7. Enabling Tez for Hive Queries

Limitations

This release of Tez does not support the following actions:

- Index creation
- Skew joins

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/hdp/<hdp_version>/hive/lib/hive-exec-1.2.1.2.
   3.0.0-2557.jar /apps/hive/install/
   ```

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;
   ```

9.8. Disabling Tez for Hive Queries

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.

9.9. 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 are configured, the default queue is used, which means that 100% of the cluster capacity is used to run Tez jobs. If queues are configured, a queue name must be configured for each YARN application.

Setting tez.queue.name in `tez-site.xml` applies to Tez applications that use that configuration file. To assign separate queues for each application, separate `tez-site.xml` files are required, or the application can pass this configuration to Tez while submitting the Tez DAG.

For example, in Hive you can use the tez.queue.name property in `hive-site.xml` to specify the queue to use for Hive-on-Tez jobs. To assign Hive-on-Tez jobs to use the "engineering" queue, 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` affects all Hive queries that read that configuration file.

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

```
bin/hive --hiveconf tez.queue.name=engineering
```

9.10. Validating 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 hadoop fs -chown hive:hdfs
   /user/test/student/student.txt hadoop fs -chmod 775
   /user/test/student/student.txt
   ```

3. Open the Hive command-line shell:

   ```
   su - $HDFS_USER
   ```

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 return successful results similar to the following:

   ```
   --------------------------------------------------------------------------------
           VERTICES      STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED
    KILLED
   --------------------------------------------------------------------------------
   Map 1 ..........   SUCCEEDED    117        117        0        0       0
       0
   Reducer 2 ......   SUCCEEDED      1          1        0        0       0
       0
   --------------------------------------------------------------------------------
   VERTICES: 02/02  [==========================>>] 100%  ELAPSED TIME: 445.02 s
   --------------------------------------------------------------------------------
   Status: DAG finished successfully in 445.02 seconds
   Time taken: 909.882 seconds
   ```

10. Installing Apache Pig

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

Complete the following tasks to install Pig:

1. Install the Pig Package [84]
2. Validate the Installation [84]

10.1. Install the Pig Package

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/hdp/current/pig-client`. Pig configuration files are placed in `/usr/hdp/current/pig-client/conf`.

10.2. Validate the Installation

To validate your installation:

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

   ```
   su - $HDFS_USER/usr/hdp/current/hadoop-client/bin/
   fs -copyFromLocal /etc/passwd passwd
   ```

2. Create a Pig script file named `/tmp/id.pig` that contains the following Pig Latin commands:

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

3. Run the Pig script:

   ```
   su - $HDFS_USER
   pig -l /tmp/pig.log /tmp/id.pig
   ```

11. Installing Apache WebHCat

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

To install WebHCat:

11.1. Install the WebHCat Package

On the WebHCat server machine, install the necessary RPMs or packages.

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install hive-hcatalog hive-webhcat
  ```

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

- For SLES:

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

11.2. Upload the Pig, Hive and Sqoop tarballs to HDFS

Upload the Pig, Hive, and Sqoop tarballs to HDFS as the $HDFS_User. The following code shows an example:

```
hdfs dfs -mkdir -p /hdp/apps/<hdp-version>/pig/
hdfs dfs -mkdir -p /hdp/apps/<hdp-version>/hive/
hdfs dfs -mkdir -p /hdp/apps/<hdp-version>/sqoop/
hdfs dfs -put /usr/hdp/<hdp-version>/pig/pig.tar.gz /hdp/apps/<hdp-version>/
pig/
hdfs dfs -put /usr/hdp/<hdp-version>/hive/hive.tar.gz /hdp/apps/<hdp-version>/
hive/
hdfs dfs -put /usr/hdp/<hdp-version>/sqoop/sqoop.tar.gz /hdp/apps/<hdp-
version>/sqoop/
hdfs dfs -chmod -R 555 /hdp/apps/<hdp-version>/pig
hdfs dfs -chmod -R 444 /hdp/apps/<hdp-version>/pig/pig.tar.gz
```

```
hdfs dfs -chmod -R 555 /hdp/apps/<hdp-version>/hive
hdfs dfs -chmod -R 444 /hdp/apps/<hdp-version>/hive/hive.tar.gz
hdfs dfs -chmod -R 555 /hdp/apps/<hdp-version>/sqoop
hdfs dfs -chmod -R 444 /hdp/apps/<hdp-version>/sqoop/sqoop.tar.gz
hdfs dfs -chown -R $HDFS_USER:$HADOOP_USER /hdp
```

11.3. 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, Hortonworks recommends that you delete them and recreate them.

Hortonworks provides a set of configuration files that represent a working WebHCat configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your WebHCat environment, complete the following steps to set up the WebHCat configuration files:

1. Execute the following 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
   hmod -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.

2. Set permissions for the WebHCat server to impersonate users on the Hadoop cluster:

   a. Create a UNIX user to run the WebHCat server.

   b. Modify the Hadoop `core-site.xml` file and set the following properties:

   Table 11.1. Hadoop core-site.xml File Properties

Variable	Value
hadoop.proxyuser.USER.groups	A comma-separated list of the UNIX groups whose users will be impersonated.

Variable	Value
hadoop.proxyuser.USER.hosts	A comma-separated list of the hosts that will run the HCatalog and JobTracker servers.

3. If you are running WebHCat on a secure cluster, create a Kerberos principal for the WebHCat server with the name **USER/host@realm**, and set the WebHCat configuration variables **templeton.kerberos.principal** and **templeton.kerberos.keytab**.

11.4. Modify WebHCat Configuration Files

Hortonworks provides a set of configuration files that represent a working WebHCat configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your WebHCat environment, complete the following steps to modify the WebHCat config files:

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

   The files are located in the `configuration_files/hive-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.

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

```
<property>
     <name>templeton.hive.properties</name>
     <value>hive.metastore.local=false,hive.metastore.uris=thrift://
$METASTORE-HOSTNAME:9083,hive.metastore.sasl.enabled=yes,hive.metastore.
execute.setugi=true,hive.metastore.warehouse.dir=/apps/hive/warehouse</
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>
```

   b. In `core-site.xml`, make sure the following properties are also set to allow WebHcat to impersonate groups and hosts:

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

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

where:

- `hadoop.proxyuser.hcat.group` is a comma-separated list of the UNIX groups whose users may be impersonated.
- `hadoop.proxyuser.hcat.hosts` is a comma-separated list of the hosts that are allowed to submit requests using **hcat**.

3. Set up the updated WebHCat configuration files.

   a. Delete any existing WebHCat configuration files:

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

   b. Copy all of the modified 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.

11.5. Set Up HDFS User and Prepare WebHCat Directories

1. Set up the WebHCat user.

   ```

   Login as $HDFS_USER
   hdfs dfs -mkdir /user/$WEBHCAT_USER
   hdfs dfs -chown -R $WEBHCAT_USER:$HDFS_USER /user/$WEBHCAT_USER
   hdfs dfs -mkdir /apps/webhcat
   ```

2. Prepare WebHCat directories on HDFS.

   ```

   hdfs dfs -copyFromLocal /usr/hdp/<hdp_version>/pig/pig.tar.gz /apps/webhcat/

   hdfs dfs -copyFromLocal /usr/hdp/<hdp_version>/hive.tar.gz /apps/webhcat/
   hdfs dfs -copyFromLocal /usr/hdp/<hdp_version>/hadoop-mapreduce/hadoop-
   streaming*.jar /apps/webhcat/
   ```

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

   ```

   hdfs fs -chown -R $WEBHCAT_USER:$HDFS_USER /apps/webhcat
   ```

```
hdfs fs -chmod -R 755 /apps/webhcat
```

where:

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

11.6. Validate the Installation

1. Start the WebHCat server and log in as $WEBHCAT_USER:

   ```
   su - $WEBHCAT_USER
   export HADOOP_HOME=/usr/hdp/current/hadoop-client
   /usr/hdp/current/hive-webhcat/sbin/webhcat_server.sh  start
   ```

2. Type the following URL into your browser:

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

   The following message is returned if your installation is valid:

   ```
   {"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.

To install Oozie:

1. Install the Oozie Package [90]
2. Set Directories and Permissions [93]
3. Set Up the Oozie Configuration Files [94]
4. Configure Your Database for Oozie [100]
5. Setting up the Sharelib [100]
6. Validate the Installation [100]

12.1. Install the Oozie Package

Prerequisites

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

   ```
   yum list oozie
   ```

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

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

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

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

Installation

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

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

   - For SLES:

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

2. On the Oozie client (typically a gateway machine), install the oozie-client package.

   For RHEL/CentOS/Oracle Linux:

```
yum install oozie-client
```

3. Update the default user profile to define a default value for OOZIE_URL.

   Typically this is done by adding a file in `/etc/profile.d` named oozie.sh. The value should be the http url of the Oozie server. Typically, http://oozie.example.com:11000/oozie for non-secure clusters or https://oozie.example.com:11443/oozie for secure clusters.

4. Install optional features: Oozie Web Console, Compression, and Drivers.

   a. Set up the Oozie lib extension directory:

      ```
      cd /usr/hdp/current/oozie-server
      ```

   b. Add the Ext 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 /usr/hdp/current/oozie-
        client/libext/
        ```

      - For SLES:

        ```
        zypper install extjs-2.2-1
        ```

        ```
        cp /usr/share/HDP-oozie/ext-2.2.zip /usr/hdp/current/oozie-
        client/libext/
        ```

   c. Add LZO JAR files:

      Note

      If you do not have an LZO JAR file, you must enable LZO compression first.

      In `/usr/hdp/current/hadoop-client/lib/hadoop-lzo-*.jar libext`

   d. Add PostgreSQL driver.

      Copy your PostgreSQL JDBC driver jar to the libext directory:

      ```
      cp /usr/share/java/postgresql-jdbc.jar /usr/hdp/current/oozie-
      client/libext/
      ```

5. Add database connector JAR files.

   If you did not already create a lib extension directory, create one now:

   ```
   cd /usr/hdp/current/oozie-client
   ```

   For MySQL:

- Copy your mysql driver jar to libext directory.

  `yum install mysql-connector-java`

  In `/usr/share/java/mysql-connector-java.jar libext/`

For Oracle

- Copy your oracle driver jar to the libext directory.

  `cp ojdbc6.jar /usr/hdp/current/oozie-client/libext/`

For Derby:

- Copy your derby driver jar to the libext directory.

  `cp derby-10-10-1.1.jar /usr/hdp/current/oozie-client/libext/`

6. Modify the `/etc/oozie/conf/oozie-env.sh` file to look like:

   From:

```
export OOZIE_CONFIG=${OOZIE_CONFIG:-/usr/hdp/current/oozie-server/conf}
export OOZIE_DATA=${OOZIE_DATA:-/var/lib/oozie/data}
export OOZIE_LOG=${OOZIE_LOG:-/var/log/oozie}
export CATALINA_BASE=${CATALINA_BASE:-/usr/hdp/current/oozie-server/oozie-
server}
export CATALINA_TMPDIR=${CATALINA_TMPDIR:-/var/tmp/oozie}
export CATALINA_PID=${CATALINA_PID:-/var/run/oozie/oozie.pid}
export OOZIE_CATALINA_HOME=/usr/lib/bigtop-tomcat
export OOZIE_BASE_URL=http://oozie.example.com:11000/oozie
```

   For secure clusters, replace the last line with the jks path and add a password to match the system configuration:

```
export OOZIE_CONFIG=${OOZIE_CONFIG:-/usr/hdp/current/oozie-server/conf}
export OOZIE_DATA=${OOZIE_DATA:-/var/lib/oozie/data}
export OOZIE_LOG=${OOZIE_LOG:-/var/log/oozie}
export CATALINA_BASE=${CATALINA_BASE:-/usr/hdp/current/oozie-server/oozie-
server}
export CATALINA_TMPDIR=${CATALINA_TMPDIR:-/var/tmp/oozie}
export CATALINA_PID=${CATALINA_PID:-/var/run/oozie/oozie.pid}
export OOZIE_CATALINA_HOME=/usr/lib/bigtop-tomcat
export OOZIE_BASE_URL=https://oozie.example.com:11443/oozie
export OOZIE_HTTPS_KEYSTORE_FILE=/etc/security/hadoop/server.jks
export OOZIE_HTTPS_KEYSTORE_PASS=changeit
```

7. Create a WAR file by running the following command as root:

```
cd /usr/hdp/current/oozie-client
bin/oozie-setup.sh prepare-war
```

8. For secure clusters, add the "-secure" parameter to the above command.

Note

If the create WAR command fails with the following error:

```
File/Dir does not exist: /usr/hdp/2.3.0.0-2800/oozie-
server/conf/ssl/server.xml
```

find the path of the SSL directory that matches oozie/tomcat-deployment/ ssl: `find / -name ssl`

For example, if that SSL path is `/grid/0/hdp/2.3.0.0-2800/oozie/tomcat-deployment/conf/ssl`, copy over that SSL directory to `usr/hdp/current/oozie-server/conf`:

```
cp -r /grid/0/hdp/2.3.0.0-2800/oozie/tomcat-deployment/
conf/ssl /usr/hdp/current/oozie-server/conf/
```

Then run `bin/oozie-setup.sh prepare-war`.

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.

Hortonworks provides a set of configuration files that represent a working Oozie configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your Oozie environment, complete the following steps to set up Oozie directories and permissions:

1. Change the permissions on the config directory by entering the following commands:

   ```
   chown root:oozie /etc/oozie
   ```

   ```
   chmod 0750 /etc/oozie
   ```

2. Hortonworks recommends that you edit and source the bash script files included in the companion files.

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

3. Run 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-confchown -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, `/hadoop/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

Hortonworks provides a set of configuration files that represent a working Oozie configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment.

If you choose to use the provided configuration files to set up your Oozie environment, complete the following steps 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. Add the following property to the `oozie-log4j.properties` file:

   ```
   log4j.appender.oozie.layout.ConversionPattern=%d{ISO8601} %5p
   %c{1}:%L - SERVER[${oozie.instance.id}] %m%n
   ```

   where ${oozie.instance.id} is automatically determined by Oozie.
3. Oozie runs a periodic purge on the shared library directory. The purge can delete libraries that are needed by jobs that started before the installation. To minimize the chance of job failures, you should extend the `oozie.service.ShareLibService.purge.interval` and `oozie.service.ShareLibService.temp.sharelib.retention.days` settings.

   Add the following content to the the `oozie-site.xml` file:

   ```
   <property>
   <name>oozie.service.ShareLibService.purge.interval</name>
   <value>1000</value>
   <description>
   ```

```
How often, in days, Oozie should check for old ShareLibs and LauncherLibs to
 purge from HDFS.
</description>
</property>

<property>
<name>oozie.service.ShareLibService.temp.sharelib.retention.days</name>
<value>1000</value>
<description>
ShareLib retention time in days.
</description>
</property>
```

4. Modify the configuration files, based on your environment and database type as described in the following sections.

12.3.1. For Derby

In the temporary directory, locate the following file and modify the properties to match your current cluster environment.

Modify the following properties in the `oozie-site.xml` file:

```
<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.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>
```

12.3.2. For MySQL

1. Install and start MySQL 5.x.

   (See "Metastore Database Requirements" and "Installing and Configuring MySQL" in Chapter 1 of this guide.)

2. Create the Oozie database and Oozie MySQL user.

   For example, using the MySQL mysql command-line tool:

```
$ mysql -u root -p
Enter password: ******

mysql> create database oozie;
Query OK, 1 row affected (0.03 sec)

mysql> grant all privileges on oozie.* to 'oozie'@'localhost' identified by
 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> grant all privileges on oozie.* to 'oozie'@'%' identified by 'oozie';
Query OK, 0 rows affected (0.03 sec)

mysql> exit
Bye
```

3. The parameter "identified by 'oozie'" sets the password for the oozie user to use "oozie". Choose a secure password for this account and add it to the `oozie-site.xml` file under oozie.service.JPAService.jdbc.password.

4. Configure Oozie to use MySQL.

```
<property>
     <name>oozie.service.JPAService.jdbc.driver</name>
     <value>com.mysql.jdbc.Driver</value>
</property>

<property>
     <name>oozie.service.JPAService.jdbc.url</name>
     <value>jdbc:mysql://localhost:3306/oozie</value>
</property>

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

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

Note

In the JDBC URL property, replace **localhost** with the hostname where MySQL is running.

5. Add the MySQL JDBC driver JAR to Oozie:

   Copy or symlink the MySQL JDBC driver JAR into the `/var/lib/oozie/` directory.

   Note

   You must manually download the MySQL JDBC driver JAR file.

12.3.3. For PostgreSQL

(See "Metastore Database Requirements" and "Installing and Configuring PostgreSQL" in Chapter 1 of this guide.)

1. Create the Oozie user and Oozie database. For example, using the PostgreSQL psql command-line tool:

```
$ psql -U postgres
Password for user postgres: *****
postgres=# CREATE ROLE oozie LOGIN ENCRYPTED PASSWORD 'oozie'
NOSUPERUSER INHERIT CREATEDB NOCREATEROLE;
CREATE ROLE

postgres=# CREATE DATABASE "oozie" WITH OWNER = oozie
ENCODING = 'UTF8'
TABLESPACE = pg_default
LC_COLLATE = 'en_US.UTF8'
LC_CTYPE = 'en_US.UTF8'
CONNECTION LIMIT = -1;
CREATE DATABASE

postgres=# \q
```

2. Configure PostgreSQL to accept network connections for the oozie user. Edit the PostgreSQL data/pg_hba.conf file as follows:

```
host oozie oozie 0.0.0.0/0 md5
```

3. Reload the PostgreSQL configuration.

```
$ sudo -u postgres pg_ctl reload -s -D /opt/PostgresSQL/8.4/data
```

4. Configure Oozie to use PostgreSQL.

   Edit the oozie-site.xml file as follows:

```
...
<property>
     <name>oozie.service.JPAService.jdbc.driver</name>
     <value>org.postgresql.Driver</value>
</property>

<property>
     <name>oozie.service.JPAService.jdbc.url</name>
     <value>jdbc:postgresql://localhost:5432/oozie</value>
</property>
```

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

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

Note

In the JDBC URL property, replace **localhost** with the hostname where PostgreSQL is running. For PostgreSQL it is unnecessary to download and install the JDBC driver separately because the driver is license-compatible and bundled with Oozie.

12.3.4. For Oracle:

(See "Metastore Database Requirements" in Chapter 1 of this guide for supported versions of Oracle. For instructions on how to install the Oracle database, see your third-party documentation.)

1. Install and start Oracle 11g.
2. Create the Oozie Oracle user.

   For example, using the Oracle SQL*Plus command-line tool:

```
$ sqlplus system@localhost
Enter password: ******
SQL> create user oozie identified by oozie default tablespace users
 temporary tablespace temp;
User created.

SQL> grant all privileges to oozie;
Grant succeeded.

SQL> exit
$
```

3. Create an Oracle database schema for Oozie to use:

   a. Set oozie.service.JPAService.create.db.schema to **true** and set oozie.db.schema.name=**oozie**.

   b. Edit the oozie-site.xml file as follows:

```
<property>
     <name>oozie.service.JPAService.jdbc.driver</name>
     <value>oracle.jdbc.driver.OracleDriver</value>
</property>

<property>
     <name>oozie.service.JPAService.jdbc.url</name>
     <value>jdbc:oracle:thin:@localhost:1521:oozie</value>
```

```
</property>

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

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

Note

In the JDBC URL property, replace **localhost** with the hostname where Oracle is running and replace **oozie** with the TNS name of the Oracle database.

4. Add the Oracle JDBC driver JAR to Oozie. Copy or symlink the Oracle JDBC driver JAR in the `/var/lib/oozie/` directory:

   `ln -s ojdbc6.jar /usr/hdp/current/oozie-server/lib`

   Note

   You must manually download the Oracle JDBC driver JAR file.

5. Modify the following properties in oozie-env.sh to match the directories created:

```
export JAVA_HOME=/usr/java/default
export OOZIE_CONFIG=${OOZIE_CONFIG:-/usr/hdp/2.3.0.0-2800/oozie/conf}
export OOZIE_DATA=${OOZIE_DATA:-/var/db/oozie}
export OOZIE_LOG=${OOZIE_LOG:-/var/log/oozie}
export CATALINA_BASE=${CATALINA_BASE:-/usr/hdp/2.3.0.0-2800/oozie}
export CATALINA_TMPDIR=${CATALINA_TMPDIR:-/var/tmp/oozie}
export CATALINA_PID=${CATALINA_PID:-/var/run/oozie/oozie.pid}
export OOZIE_CATALINA_HOME=/usr/lib/bigtop-tomcat
export OOZIE_CLIENT_OPTS="${OOZIE_CLIENT_OPTS} -Doozie.connection.retry.
count=5"
export CATALINA_OPTS="${CATALINA_OPTS} -Xmx2048m -XX:MaxPermSize=256m"
export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
```

6. 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;
```

7. Copy all the config files to $OOZIE_CONF_DIR directory.

8. 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.

12.4. Configure Your Database for Oozie

Most of the configuration steps for your database for Oozie were performed in the previous section. However, you still need to create the sqlfile to create the tables in the database and validate that the configuration and jdbc are correct.

1. Run the following command as oozie user:

   ```
   /usr/hdp/current/oozie-server/bin/ooziedb.sh create -sqlfile /
   tmp/oozie.sql -run
   ```

12.5. Setting up the Sharelib

1. Run the following command as oozie user:

   ```
   cd /tmp
   tar xzf /usr/hdp/${hdp_version}/oozie/oozie-sharelib.tar.gz
   hadoop fs -put share /user/oozie
   ```

12.6. Validate the Installation

1. Start the Oozie server:

   ```
   su -l oozie -c "/usr/hdp/current/oozie-server/bin/oozied.sh
   start"
   ```

   where oozie is the $OOZIE_User.

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

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

3. Access the Oozie server with the Oozie client:

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

The following message is returned if your installation is valid:

```
System mode: NORMAL
```

Next Steps

For example workflow templates, download the companion files, and use \oozie_workflows.

See the Apache website for more information about Apache Oozie.

13. Installing Apache Ranger

Apache Ranger delivers a comprehensive approach to security for a Hadoop cluster. It provides central security policy administration across the core enterprise security requirements of authorizaiton, auditing and data protection.

This chapter describes the manual installation process Apache Ranger and the Ranger plug-ins in a LINUX Hadoop environment. It includes information about the following steps:

For information on the different tasks you can perform with Apache Ranger, refer to the Ranger User Guide. For information about installing Ranger over Ambari, see Installing Ranger Over Ambari 2.0.

13.1. Installation Prerequisites

Before beginning Ranger installation, make sure the following software is already installed:

- JDK 7 or above (available from the Oracle Java Download site)
- Supported Operating Systems:
  - 64-bit CentOS 6
  - 64-bit CentOS 7
  - 64-bit Red Hat Enterprise Linux (RHEL) 6
  - 64-bit Oracle Linux 6
  - 64-bit SUSE Linux Enterprise Server (SLES) 11, SP3/SP4
  - 64-bit Debian 6
  - Windows Server 2008, 2012
- Supported Databases:
  - MySQL v. 5.6 or above
  - PostgreSQL 8.4.2
  - SQL Server 2012
  - Oracle DB 11G or above (Oracle LINUX 6)

If the database server is not installed at the same host, Ranger services need to have access to the database server host.

Note

If you are installing over an HDP-2.2 version of Ranger, you should review the data in the xa_access_audit table in the Ranger Audit Database. Truncate historical data to reduce install time, especially if you have a lot of data in xa_access_audit table.

13.2. Installing Policy Manager

This section describes how to perform the following administrative tasks:

1. Configure and install the Ranger Policy Manager
2. Start the Policy Manager service

13.2.1. Install the Ranger Policy Manager

1. Make sure the HDP 2.3 repository is added to your site's list of yum repositories.

   If it has not yet been added, add it now by performing the following steps:

   - For RHEL/Centos6/Oracle LINUX 6:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.3.0.0/
     hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - For Ubuntu 12.04:

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/
     GA/2.3.0.0/hdp.list -O /etc/apt/sources.list.d/hdp.list
     ```

   - For Debian 6:

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDP/debian6/2.x/
     GA/2.3.0.0/hdp.list -O /etc/apt/sources.list.d/hdp.list
     ```

2. Find the Ranger Policy Admin software:

   ```
   yum search ranger
   ```

3. Install the Ranger Policy Admin software:

   ```
   yum install ranger_<version>
   ```

4. In the Ranger Policy Administration installation directory, update the `install.properties` file:

   - Go to the installation directory:

     ```
     cd /usr/hdp/<version>/ranger-admin/
     ```

   - Edit the following install.properties entries:

Table 13.1. install.properties Entries

Configuration Property	Default/Example Value	Required?
Ranger Policy Database		
DB_FLAVOR Specifies the type of database used (MYSQL,ORACLE,POSTGRES,MSSQL)	MYSQL (default)	Y
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
db_root_user database username who has privileges for creating database schemas and users	root (default)	Y
db_root_password database password for the "db_root_user"	rootPassW0Rd	Y
db_host Hostname of the Ranger policy database server	localhost	Y
db_name Ranger Policy database name	ranger (default)	Y
db_user db username used for performing all policy mgmt operation from policy admin tool	rangeradmin (default)	Y
db_password database password for the "db_user"	RangerAdminPassW0Rd	Y
Ranger Audit Database		
audit_db_name Ranger audit database name - This can be a different database in the same database server mentioned above	ranger_audit (default)	Y
audit_db_user Ranger audit database name - This can be a different database in the same database server mentione	rangerlogger (default)	Y
audit_db_password database password for the "audit_db_user"	RangerLoggerPassW0Rd	Y
Policy Admin Tool Config		
policymgr_external_url URL used within Policy Admin tool when a link to its own page is generated in the Policy Admin Tool website	*http://localhost:6080* (default) *http://myexternalhost.xasecure.net:6080N*	
policymgr_http_enabled Enables/disables HTTP protocol for downloading policies by Ranger plug-ins	true (default)	Y
unix_user UNIX user who runs the Policy Admin Tool process	ranger (default) (default)	Y
unix_group UNIX group associated with the UNIX user who runs the Policy Admin Tool process	ranger (default)	Y
Policy Admin Tool Authentication		
authentication_method	none (default)	Y

Configuration Property	Default/Example Value	Required?
Authentication Method used to log in to the Policy Admin Tool. NONE – only users created within the Policy Admin Tool may log in UNIX – allows UNIX userid authentication using the UNIX authentication service (see below) LDAP – allows Corporate LDAP authentication (see below) ACTIVE_DIRECTORY – allows authentication using an Active Directory		
UNIX Authentication Service		
remoteLoginEnabled Flag to enable/disable remote Login via Unix Authentication Mode	true (default)	Y, if UNIX authentication_method is selected
authServiceHostName Server Name (or ip-addresss) where ranger-usersync module is running (along with Unix Authentication Service)	localhost (default) myunixhost.domain.com	Y, if UNIX authentication_method is selected
authServicePort Port Number where ranger-usersync module is running Unix Authentication Service	5151 (default)	Y, if UNIX authentication_method is selected
LDAP Authentication		
xa_ldap_url URL for the LDAP service	ldap://<ldapServer>:389	Y, if LDAP authentication_method is selected
xa_ldap_userDNpattern LDAP DN Pattern used to uniquely locate the login user	uid={0},ou=users,dc=xasecure,dc=net	Y, if LDAP authentication_method is selected
xa_ldap_groupSearchBase LDAP Base node location to get all groups associated with login user	ou=groups,dc=xasecure,dc=net	Y, if LDAP authentication_method is selected
xa_ldap_groupSearchFilter LDAP search filter used to retrieve groups for the login user	(member=uid={0},ou=users, dc=xasecure,dc=net)	Y, if LDAP authentication_method is selected
xa_ldap_groupRoleAttribute Attribute used to retrieve the group names from the group search filters	cn	Y, if LDAP authentication_method is selected
Active Directory Authentication		
xa_ldap_ad_domain Active Directory Domain Name used for AD login	xasecure.net	Y, if ACTIVE_DIRECTORY authentication_method is selected
xa_ldap_ad_url Active Directory LDAP URL for authentication of user	ldap://ad.xasecure.net:389	Y, if ACTIVE_DIRECTORY authentication_method is selected

5. Check the JAVA_HOME environment variable. If it has not yet been set, enter:

```
export JAVA_HOME=<path of installed jdk version folder>
```

13.2.2. Install the Ranger Policy Administration Service

To install the Ranger Policy Administration service, run the following commands:

```
cd /usr/hdp/<version>/ranger-admin
```

```
./setup.sh
```

13.2.3. Start the Ranger Policy Administration Service

To start the Ranger Policy Administration service, enter the following command:

```
service ranger-admin start
```

To verify that the service started, visit the external URL specified in install.properties in browser; for example:

```
http://<host_address>:6080/
```

Note

The default user is "admin" with a password of "admin". After login, change the password for "admin".

13.2.4. Configuring the Ranger Policy Administration Authentication Mode

The Ranger service also enables you to configure the authentication method the Ranger Policy Administration component uses to authenticate users. There are three different authentication methods supported with Ranger, which include:

- Active Directory (AD)
- LDAP
- UNIX

Depending on which authentication method you choose, you will need to modify the following sample values in the `install.properties` file:

Active Directory

- authentication_method=ACTIVE_DIRECTORY
- xa_ldap_ad_domain= example.com
- xa_ldap_ad_url=ldap://127.0.0.1:389
- xa_ldap_ad_base_dn=DC=example,DC=com
- xa_ldap_ad_bind_dn=CN=Administrator,CN=Users,DC=example,DC=com
- xa_ldap_ad_bind_password=PassW0rd
- xa_ldap_ad_referral=ignore, follow or throw. Default is follow.

LDAP

- authentication_method=LDAP
- xa_ldap_url=LDAP server URL (e.g. ldap://127.0.0.1:389)
- xa_ldap_userDNpattern=uid={0},ou=users,dc=example,dc=com
- xa_ldap_groupSearchBase=dc=example,dc=com
- xa_ldap_groupSearchFilter=(member=cn={0},ou=users,dc=example,dc=com
- xa_ldap_groupRoleAttribute=cn
- xa_ldap_base_dn=dc=example,dc=com
- xa_ldap_bind_dn=cn=ldapadmin,ou=users,dc=example,dc=com
- xa_ldap_bind_password=PassW0rd
- xa_ldap_referral=ignore, follow, or throw. Default is follow.

UNIX

- authentication_method=UNIX
- remoteLoginEnabled=true
- authServiceHostName= an address of the host where the UNIX authentication service is running.
- authServicePort=5151

13.2.5. Configuring Ranger Policy Administration High Availability

If you would like to enable high availability for the Ranger Policy Administration component, you can configure the component by following the steps listed below.

1. Install the Ranger Admin component on the hosts you wish to use.

   Note

   Make sure you use the same configuration and policy database settings for each host, otherwise, you will be unable to configure HA for the hosts.

2. Configure a load balancer to balance the loads among the various Ranger Admin instances and take note of the load balancer URL.

   Note

   It is out of the scope in this document to describe the steps you need to follow in order to install and configure a load balancer.

3. Update the Policy Manager external URL in all Ranger Admin clients (Ranger UserSync and Ranger plug-ins) to point to the load balancer URL.

13.3. Installing UserSync

To install Ranger UserSync and start the service, do the following:

1. Find the Ranger UserSync software:

   ```
   yum search usersync
   ```

   or

   ```
   yum list | grep usersync
   ```

2. Install Ranger UserSync:

   Note

   Make sure the database on which Ranger will be installed is up and running.

   ```
   yum install ranger_<version>-usersync.x86_64
   ```

3. At the Ranger UserSync installation directory, update the following properties in the `install.properties` file:

Table 13.2. Properties to Update in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	*http://policymanager.xasecure.net:6080*	Y
User Group Source Information		
SYNC_SOURCE Specifies where the user/group information is extracted to be put into Ranger database. unix - get user information from /etc/passwd file and gets group information from /etc/group file ldap - gets user information from LDAP service (see below for more information)	unix	N
SYNC_INTERVAL Specifies the interval (in minutes) between synchronization cycle. Note, the 2nd sync cycle will NOT start until the first sync cycle is COMPLETE.	5	N
UNIX user/group Synchronization		
MIN_UNIX_USER_ID_TO_SYNC UserId below this parameter values will not be synchronized to Ranger user database	300 (Unix default), 1000 (LDAP default)	Mandatory if SYNC_SOURCE is selected as unix
LDAP user/group synchronization		
SYNC_LDAP_URL URL of source ldap	ldap://ldap.example.com:389	Mandatory if SYNC_SOURCE is selected as ldap

Configuration Property Name	Default/Example Value	Required?
SYNC_LDAP_BIND_DN ldap bind dn used to connect to ldap and query for users and groups	cn=admin,ou=users,dc=hadoop, dc=apache,dc-org	Mandatory if SYNC_SOURCE is selected as ldap
SYNC_LDAP_BIND_PASSWORD ldap bind password for the bind dn specified above	LdapAdminPassW0Rd	Mandatory if SYNC_SOURCE is selected as ldap
CRED_KEYSTORE_FILENAME Location of the file where crypted password is kept	*/usr/lib/xausersync/.jceks/ xausersync.jceks* (default) */ etc/ranger/usersync/.jceks/ xausersync.jceks*	Mandatory if SYNC_SOURCE is selected as ldap
SYNC_LDAP_USER_SEARCH_BASE Search base for users	ou=users,dc=hadoop,dc=apache, dc=org	Mandatory if SYNC_SOURCE is selected as ldap
SYNC_LDAP_USER_SEARCH_SCOPE Search scope for the users, only base, one, and sub are supported values	sub (default)	N
SYNC_LDAP_USER_OBJECT_CLASS objectclass to identify user entries	person (default)	N (defaults to person)
SYNC_LDAP_USER_SEARCH_FILTER Optional additional filter constraining the users selected for syncing	(dept=eng)	N (defaults to an empty string)
SYNC_LDAP_USER_NAME _ATTRIBUTE Attribute from user entry that would be treated as user name	cn (default)	N (defaults to cn)
SYNC_LDAP_USER_GROUP_NAME _ATTRIBUTE attribute from user entry whose values would be treated as group values to be pushed into Policy Manager database. You can provide multiple attribute names separated by comma	memberof,ismemberof (default)	N (defaults to memberof, ismemberof)
User Synchronization		
unix_user UNIX User who runs the ranger-usersync process	ranger (default)	Y
unix_group UNIX group associated with Unix user who runs the ranger-usersync process	ranger (default)	Y
SYNC_LDAP_USERNAME_CASE _CONVERSION Convert all username to lower/upper case none - no conversation will be done. Kept as it is in the SYNC_SOURCE lower - convert it to lower case when saving it to ranger db upper - convert it to upper case when saving it to ranger db	lower (default)	N (defaults to lower)
SYNC_LDAP_GROUPNAME_CASE _CONVERSION Convert all username to lower/upper case none - no conversation will be done. Kept as it is in the SYNC_SOURCE lower - convert it to lower case when saving it to ranger db upper - convert it to upper case when saving it to ranger db	lower (default)	N (defaults to lower)
logdir Location of the log directory were the usersync logs are stored	logs (default)	Y

4. Set the Policy Manager URL to *http://<ranger-admin-host>:6080*

5. Check the JAVA_HOME environment variable. If JAVA_HOME has not yet been set, enter:

   ```
   export JAVA_HOME=<path of installed jdk version folder>
   ```

6. Install the Ranger UserSync service:

   ```
   cd /usr/hdp/<version>/ranger-usersync
   ```

   ```
   ./setup.sh
   ```

7. Start the Ranger UserSync service:

   ```
   service ranger-usersync start
   ```

8. To verify that the service was successfully started, wait 6 hours for LDAP and AD to synchronize, then do the following:

   - Go to

     ```
     http://<ranger-admin-host>:6080
     ```

   - Click the Users/Group tab. See if users and groups are synchronized.

   - Add a UNIX/LDAP/AD user, then check for the presence of that user in the Ranger Admin tab.

13.4. Installing Ranger Plug-ins

The following sections describe how to install Ranger plug-ins.

Note

To ensure that you are installing the HDP version of the plug-ins instead of the Apache version, make sure you enter the following commands when installing each plug-in:

- For CentOS and RHEL:

  ```
  yum install ranger_ <version_number>
  ```

- For SLES:

  ```
  zypper -n --no-gpg-checks install --auto-agree-with-licenses
   ranger_ <version_number>
  ```

- For Debian/Ubuntu:

  ```
  apt-get install <version_number>
  ```

13.4.1. Installing the Ranger HDFS Plug-in

The Ranger HDFS plug-in helps to centralize HDFS authorization policies.

This section describes how to create an HDFS repository and install the HDFS plug-in.

Install the HDFS Plug-in

1. Create an HDFS repository in the Ranger Policy Manager. To do this, complete the HDFS Create Repository screen, as described in the HDFS Repository Configuration section of the *Hortonworks Data Platform Ranger User Guide*.

   Make a note of the name you gave to this repository; you will need to use it again during HDFS plug-in setup.

2. At all servers where NameNode is installed, install the HDFS plug-in by following the steps listed below:

   a. Go to the home directory of the HDFS plug-in:

   ```
   cd /usr/hdp/<version>/ranger-hdfs-plugin
   ```

   b. Edit the following HDFS-related properties in the install.properties file:

Table 13.3. Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	http://policymanager.xasecure.net:6080	Y
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	hadoopdev	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging.	FALSE (default), TRUE	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD Database password associated with the above database user - for db audit logging	rangerlogger	Y
HDFS Audit		

Configuration Property Name	Default/Example Value	Required?
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs		Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%	Y
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) /var/log/%app-type%/ audit	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR%app-type %/audit/archive (format) /var/log/ %app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write to a new file at a rollover interval specified here	86400	Y
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS hdfs audit log open() call is failed, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Local audit log file is rotated to write to a new file at a rollover interval specified here	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files that will be kept in the archive directory	10	Y
SSL Information (https connectivity to Policy Admin Tool)		
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as	/etc/hadoop/conf/ranger-plugin-keystore.jks (default)	Only if SSL is enabled

Configuration Property Name	Default/Example Value	Required?
it is - do not set as EMPTY if SSL not used		
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	/etc/hadoop/conf/ranger-plugin-truststore.jks (default)	Only if SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled

3. To enable the HDFS plug-in, run the following commands:

```
cd /usr/hdp/<version>/ranger-hdfs-plugin
```

```
./enable-hdfs-plugin.sh
```

4. To confirm that installation and configuration are complete, go to the Audit Tab of the Ranger Admin Console and check Plugins. You should see HDFS listed there.

13.4.2. Installing the Ranger YARN Plug-in

This section describes how to install and enable the Ranger YARN plug-in.

1. The Ranger YARN plug-in is automatically installed when YARN is installed. You can verify this plug-in is present by using the following command:

```
rpm -qa | grep yarn-plugin
ranger_2_3_0_0_2437-yarn-plugin-0.5.0.2.3.0.0-2437.el6.x86_64
```

2. Navigate to `/usr/hdp/<version>/ranger-yarn-plugin`.

```
cd /usr/hdp/<version>/ranger-yarn-plugin
```

3. Edit the following entries in the `install.properties` file.

Table 13.4. Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	http://<FQDN of ranger admin host>:6080	Y

Configuration Property Name	Default/Example Value	Required?
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	yarndev	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging.	FALSE (default), TRUE	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD Database password associated with the above database user - for db audit logging	rangerlogger	Y
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs		Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%	Y
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) /var/log/%app-type%/ audit	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR%app-type %/audit/archive (format) /var/log/ %app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write	86400	Y

Configuration Property Name	Default/Example Value	Required?
to a new file at a rollover interval specified here		
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS hdfs audit log open() call is failed, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Local audit log file is rotated to write to a new file at a rollover interval specified here	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files that will be kept in the archive directory	10	Y
SSL Information (https connectivity to Policy Admin Tool)		
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	/etc/hadoop/conf/ranger-plugin-keystore.jks (default)	Only if SSL is enabled
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	/etc/hadoop/conf/ranger-plugin-truststore.jks (default)	Only if SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled

4. Enable the YARN plug-in by running the following commands:

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
```

or

```
export JAVA_HOME=/usr/lib/jvm/java-1.8.0-openjdk-amd64

./enable-yarn-plugin.sh
```

5. Make sure HADOOP_YARN_HOME and HADOOP_LIBEXEC_DIR are set.

   ```
   export HADOOP_YARN_HOME=/usr/hdp/current/hadoop-yarn-nodemanager/
   export HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec/
   ```

6. Enter the following commands to stop/start the ResourceManager on all of your Resource Manager hosts.

   ```
   su yarn -c "/usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-daemon.sh
    stop resourcemanager"
   su yarn -c "/usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-daemon.sh
    start resourcemanager"
   ps -ef | grep -i resourcemanager
   ```

7. Enter the following command to stop/start the NodeManager on all of your NodeManager hosts.

   ```
   su yarn -c "/usr/hdp/current/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh
    stop nodemanager"
   su yarn -c "/usr/hdp/current/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh
    start nodemanager"
   ps -ef | grep -i nodemanager
   ```

8. Create the default repo for YARN with the proper configuration specifying the same repository name as in step 3.

9. You can verify the plug-in is communicating to Ranger admin via the Audit/plugins tab.

13.4.3. Installing the Ranger Kafka Plug-in

This section describes how to install and enable the Ranger Kafka plug-in.

1. The Ranger Kafka plug-in is automatically installed when Kafka is installed. You can verify this plug-in is present by using the following command:

   ```
   rpm -qa | grep kafka-plugin
   ranger_2_3_0_0_2437-kafka-plugin-0.5.0.2.3.0.0-2437.el6.x86_64
   ```

2. Navigate to `/usr/hdp/<version>/ranger-kafka-plugin`.

   ```
   cd /usr/hdp/<version>/ranger-kafka-plugin
   ```

3. Edit the following entries in the `install.properties` file.

Table 13.5. Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
COMPONENT_INSTALL_DIR_NAME	/usr/hdp/2.3.0.0-2437/kafka	Y
POLICY_MGR_URL URL for policy admin	http://<FQDN of ranger admin host>:6080	Y

Configuration Property Name	Default/Example Value	Required?
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	kafkadev	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging.	FALSE (default), TRUE	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD Database password associated with the above database user - for db audit logging	rangerlogger	Y
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs		Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%	Y
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	hdfs:// __REPLACE__NAME_NODE_HOST:8020/ (format) /var/log/%app-type%/ audit	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR%app-type %/audit/archive (format) /var/log/ %app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write	86400	Y

Configuration Property Name	Default/Example Value	Required?
to a new file at a rollover interval specified here		
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS hdfs audit log open() call is failed, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Local audit log file is rotated to write to a new file at a rollover interval specified here	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files that will be kept in the archive directory	10	Y
SSL Information (https connectivity to Policy Admin Tool)		
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	/etc/hadoop/conf/ranger-plugin-keystore.jks (default)	Only if SSL is enabled
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	/etc/hadoop/conf/ranger-plugin-truststore.jks (default)	Only if SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used	none (default)	Only if SSL is enabled

4. Enable the Kafka plug-in by running the following commands:

```
export JAVA_HOME=/usr/lib/jvm/java-1.7.0-openjdk-amd64
./enable-kafka-plugin.sh
```

5. Enter the following commands to stop/start the Kafka service.

   ```
   su kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
   su kafka -c "/usr/hdp/current/kafka-broker/bin/kafka start"
   ```

6. Create the default repo for Kafka with the proper configuration specifying the same repository name as in step 3.

7. You can verify the plug-in is communicating to Ranger admin via the Audit/plugins tab.

8. If the plug-in is not able to communicate with Ranger admin, check the property `authorizer.class.name` in `/usr/hdp/2.3.0.0-2437/kafka/config/server.properties`. The value of the authorizer.class.name should be org.apache.ranger.authorization.kafka.authorizer.RangerKafkaAuthorizer.

13.4.4. Installing the Ranger HBase Plug-in

The Ranger HBase Plug-in integrates with HBase to enforce authorization policies.

This section describes how to install the HBase plug-in:

1. Create an HBase repository
2. Install the HBase plug-in and configure related HBase properties
3. Enable the HBase plug-in
4. Restart HBase

Install the HBase Plug-in

1. Create an HBase repository in the Ranger Policy Manager. To do this, complete the HBase Create Repository screen, as described in the HBase Repository Configuration section of the *Apache Ranger User Guide*.

   Make a note of the name you gave to this repository; you will use it again during HBase plug-in setup.

2. At all servers where the HBase Master and RegionServers are installed, install and configure the HBase plug-in, as follows:

   a. Go to the home directory of the HBase plug-in:

      ```
      cd /usr/hdp/<version>/ranger-hbase-plugin
      ```

   b. Edit the following HBase-related properties in the `install.properties` file:

      Table 13.6. HBase Properties to Edit in the `install.properties` file

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	*http://policymanager.xasecure.net:6080*	Y
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	hbasedev	Y
Audit Database		

Configuration Property Name	Default/Example Value	Required?
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging. *Note*: If this property is set to FALSE, Ranger will not store audit logs in the audit DB, and audit logs will not be visible in the Ranger UI. If you would like to access audit logs from the UI, set this value to TRUE.	FALSE (default)	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD Database password associated with the above database user - for db audit logging	rangerlogger	Y
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs	TRUE	Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	*hdfs:// __REPLACE__NAME_NODE_HOST:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd% (format) hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%*	Y
XAAUDIT.HDFS.LOCAL _BUFFER_DIRECTORY Local directory where the audit log will be saved for intermediate storage	__REPLACE__LOG_DIR/%app-type %/audit (format) /var/tmp/%app-type%/audit	Y
XAAUDIT.HDFS.LOCAL _ARCHIVE_DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR/%app-type %/audit/archive (format) /var/ tmp/%app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE HDFS audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS	900	Y

Configuration Property Name	Default/Example Value	Required?
HDFS audit log file writes are flushed to HDFS at regular flush interval		
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS HDFS audit log file is rotated to write to a new file at a rollover interval specified here	86400	Y
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS If HDSF audit log open() call fails, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL _BUFFER_FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Interval that local audit log file writes are flushed to filesystem	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Interval that local audit log file is rolled over (rotated to write to a new file)	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files will be kept in the archive directory	10	Y
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Used only if SSL is enabled between Policy Admin Tool and Plugin. If SSL is not enabled, leave the default value as it is (should not be set as EMPTY).	*/etc/hbase/conf/ranger-plugin-keystore.jks* (default)	Y, if SSL is enabled
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Used only if SSL is enabled between Policy Admin Tool and Plugin. If SSL is not Enabled, leave the default value as it is (should not be set as EMPTY).	myKeyFilePassword (default)	Y, if SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Used only if SSL is enabled between Policy Admin Tool and Plugin. If SSL is not enabled, leave the default value as it is (should not be set as EMPTY).	*/etc/hbase/conf/ranger-plugin-truststore.jks* (default)	Y, if SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Used only if SSL is enabled between Policy Admin Tool and Plugin. If SSL is not Enabled, leave the default value as it is (should not be set as EMPTY).	changeit (default)	Y, if SSL is enabled
HBase GRANT/REVOKE Commands		

Configuration Property Name	Default/Example Value	Required?
UPDATE_XAPOLICIES_ON_GRANT_REVOKE Provide ability for XAAgent to update the policies based on the GRANT/REVOKE commands from the HBase client	TRUE (default)	Y

3. To enable the HBase plug-in, enter the following commands:

```
cd /usr/hdp/<version>l/ranger-hbase-plugin
```

```
./enable-hbase-plugin.sh
```

4. Restart HBase.

5. To confirm that the HBase plug-in installation and configuration are complete, go to the Audit Tab of the Ranger Admin Console and check Plugins. You should see HBase listed there.

13.4.5. Installing the Ranger Hive Plug-in

The Ranger Hive plug-in integrates with Hive to enforce authorization policies.

Note

The Ranger plugin for HIve only needs to be set up for HiveServer2. For Hive clients, it is recommended that you protect data using HDFS policies in Ranger. Do not install or set up Ranger plugins on individual Hive client machines.

This section describes how to install the Ranger Hive plug-in:

1. Create a Hive repository.

2. Install the Hive plug-in and configure related Hive properties.

3. Enable the Hive plug-in.

4. Restart Hive.

Install the Hive Plug-in

1. Create a Hive repository. To create the Hive repository, complete the Hive Create Repository screen as described in the Hive Repository Configuration section of the *Apache Ranger User Guide*.

   Make a note of the name you gave to this repository; you will need to use it again during Hive plug-in setup.

2. At the server where HiveServer2 is installed, install the Hive plug-in:

   • Go to the home directory of the Hive plug-in:

   ```
   cd /usr/hdp/<version>/ranger-hive-plugin
   ```

   • Edit the following Hive-related properties in the install.properties file:

Table 13.7. Hive-Related Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	*http://policymanager.xasecure.net:6080*	Y
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	hivedev	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging. *Note*: If this property is set to FALSE, Ranger will not store audit logs in the audit DB, and audit logs will not be visible in the Ranger UI. If you would like to access audit logs from the UI, set this value to TRUE.	FALSE (default) TRUE	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD database password associated with the above database user - for db audit logging	rangerlogger	Y
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging.If the hdfs audit logging is turned off, it will not log any access control to hdfs		Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	*hdfs://__REPLACE__NAME_NODE_HOST:8020/ranger/audit/%app-type%/%time:yyyyMMdd%* (format) *hdfs://namenode.mycompany.com:8020/ranger/audit/%app-type%/%time:yyyyMMdd%*	Y

Configuration Property Name	Default/Example Value	Required?
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	__REPLACE__LOG_DIR/%app-type %/audit (format) /var/tmp/%app-type%/audit	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR/%app-type %/audit (format) /var/tmp/%app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write to a new file at a rollover interval specified here	86400	Y
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS If hdfs audit log open() call is failed, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60	Y
X**AAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS** Local audit log file is rotated to write to a new file at a rollover interval specified here	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files that will be kept in the archive directory	10	Y
SSL Information (https connectivity to Policy Admin Tool)		
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	*/etc/hive/conf/ranger-plugin-keystore.jks* (default)	If SSL is enabled
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	none (default)	If SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying	*/etc/hive/conf/ranger-plugin-truststore.jks* (default)	If SSL is enabled

Configuration Property Name	Default/Example Value	Required?
SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.		
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	none (default)	If SSL is enabled
Hive GRANT/REVOKE Command Handling		
UPDATE_XAPOLICIES_ON_GRANT_REVOKE Provide ability for XAAgent to update the policies based on the grant/revoke commands from the Hive beeline client	TRUE (default)	Y

3. To enable the Hive plug-in, enter the following commands:

```
cd /usr/hdp/<version>/ranger-hive-plugin
```

```
./enable-hive-plugin.sh
```

4. Restart Hive.

5. To confirm that the Hive plug-in installation and configuration are complete, go to the Audit Tab of the Ranger Admin Console and check Plugins. You should see Hive listed there.

13.4.6. Installing the Ranger Knox Plug-in

The Ranger Knox plug-in integrates with Knox to enforce authorization policies.

This section describes how to install the Knox plug-in:

1. Create a Knox repository.

2. Install the Knox plug-in and configure related Hive properties.

3. Enable the Knox plug-in.

4. Restart Knox.

Instructions assume that Knox has already been installed, as described in "*Installing Knox*."

Install the Knox Plug-in

1. Create a Knox repository. To do this, complete the Knox Create Repository screen as described in the Knox Repository Configuration section of the *Apache Ranger User Guide*.

2. Set the URL to *https://knox_host:8443/gateway/admin/api/v1/topologies*, where knox_host is the full-qualified name of your Knox host machine.

3. Make a note of the name you gave to this repository; you will need to use it again during Knox plug-in setup.

4. At all servers where Knox Gateway is installed, install the Knox plug-in:

   a. Go to the home directory of the Knox plug-in:

   ```
   cd /usr/hdp/<version>/ranger-knox-plugin
   ```

   b. Edit the following Knox-related properties in the `install.properties` file:

Table 13.8. Knox-Related Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Mandatory?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	*http://policymanager.xasecure.net:6080*	Y
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	knoxdev	Y
Knox Component Installation		
KNOX_HOME Home directory where Knox software is installed	*/usr/hdp/current/knox*	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	*/usr/share/java/mysql-connector-java.jar*	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging. *Note*: If this property is set to FALSE, Ranger will not store audit logs in the audit DB, and audit logs will not be visible in the Ranger UI. If you would like to access audit logs from the UI, set this value to TRUE.	true	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE)	MYSQL	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD database password associated with the	rangerlogger	Y

Configuration Property Name	Default/Example Value	Mandatory?
above database user - for db audit logging		
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs.		Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	*hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%*	Y
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	*/var/tmp/%app-type%/audit*	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	*/var/tmp/%app-type%/audit/archive*	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write to a new file at a rollover interval specified here	86400	Y
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS If hdfs audit log open() call is failed, it will be re-tried at this interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Local audit log file is rotated to write to a new file at a rollover interval specified here	600	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files will be kept in the archive directory	10	Y
SSL (https connectivity to Policy Admin Tool)		
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy	*/etc/knox/conf/ranger-plugin-keystore.jks*	If SSL is enabled

Configuration Property Name	Default/Example Value	Mandatory?
Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.		
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	myKeyFilePassword	If SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	*/etc/knox/conf/ranger-plugin-truststore.jks*	If SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	changeit	If SSL is enabled

5. To enable the Knox plug-in, enter the following commands:

```
cd /usr/hdp/<version>/ranger-knox-plugin
```

```
./enable-knox-plugin.sh
```

6. Restart the Knox Gateway.

7. To confirm that the Knox plug-in installation and configuration are complete, go to the Audit Tab of the Ranger Admin Console and check Plugins. You should see Knox listed there.

13.4.7. Installing the Ranger Storm Plug-in

The Ranger Storm plug-in integrates with Storm to enforce authorization policies.

This section describes how to perform the following administrative tasks: It assumes that Storm has already been installed, as described earlier in this guide.

1. Create a Storm repository.

2. Install the Storm plug-in and configure related Storm properties.

3. Enable the Storm plug-in.

4. Restart Storm.

Install the Storm Plug-in

1. Create a Storm repository, as described in the Storm Repository Configuration section of the *Apache Ranger User Guide*.

   Make a note of the name you gave to this repository; you will need to use it again during Storm plug-in setup.

2. On the Nimbus server, install the Storm plug-in:

   a. Go to the home directory of the Storm plug-in:

   ```
   cd /usr/hdp/<version>/ranger-storm-plugin
   ```

   b. Edit the following Storm-related properties in the `install.properties` file:

Table 13.9. Storm-Related Properties to Edit in the `install.properties` file

Configuration Property Name	Default/Example Value	Mandatory?
Policy Admin Tool		
POLICY_MGR_URL URL for policy admin	*http://policymanager.xasecure.net:6080*	Y
REPOSITORY_NAME The repository name used in Policy Admin Tool for defining policies	stormdev	Y
Audit Database		
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging. **Note**: If this property is set to FALSE, Ranger will not store audit logs in the audit DB, and audit logs will not be visible in the Ranger UI. If you would like to access audit logs from the UI, set this value to TRUE.	false (default) true	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE, PostgreSQL 8.4.2, SQL Server 2012)	MYSQL (default)	Y
XAAUDIT.DB.HOSTNAME Hostname of the audit database server	localhost	Y
XAAUDIT.DB.DATABASE_NAME Audit database name	ranger_audit	Y
XAAUDIT.DB.USER_NAME Username used for performing audit log inserts (should be same username used in the ranger-admin installation process)	rangerlogger	Y
XAAUDIT.DB.PASSWORD Database password associated with	rangerlogger	Y

Configuration Property Name	Default/Example Value	Mandatory?
the above database user - for db audit logging		
HDFS Audit		
XAAUDIT.HDFS.IS_ENABLED Flag to enable/disable hdfs audit logging. If the hdfs audit logging is turned off, it will not log any access control to hdfs.	false	Y
XAAUDIT.HDFS.DESTINATION _DIRECTORY HDFS directory where the audit log will be stored	*hdfs:// __REPLACE__NAME_NODE_HOST:8020/ ranger/audit/%app-type%/ %te:yyyyMMdd%* (format) *hdfs:// namenode.mycompany.com:8020/ ranger/audit/%app-type%/ %time:yyyyMMdd%*	Y
XAAUDIT.HDFS.LOCAL_BUFFER _DIRECTORY Local directory where the audit log will be saved for intermediate storage	__REPLACE__LOG_DIR/%app-type %/audit (format) /var/log/%app-type%/audit	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _DIRECTORY Local directory where the audit log will be archived after it is moved to hdfs	__REPLACE__LOG_DIR/%app-type %/audit/archive (format) /var/log/ %app-type%/audit/archive	Y
XAAUDIT.HDFS.DESTINATION_FILE hdfs audit file name (format)	%hostname%-audit.log (default)	Y
XAAUDIT.HDFS.DESTINATION _FLUSH_INTERVAL_SECONDS hdfs audit log file writes are flushed to HDFS at regular flush interval	900 (default)	Y
XAAUDIT.HDFS.DESTINATION _ROLLOVER_INTERVAL_SECONDS hdfs audit log file is rotated to write to a new file at a rollover interval specified here	86400 (default)	Y
XAAUDIT.HDFS.DESTINATION _OPEN_RETRY_INTERVAL_SECONDS If hdfs audit log open() call is failed, it will be re-tried at this interval	60 (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FILE Local filename used to store in audit log (format)	%time:yyyyMMdd-HHmm.ss%.log (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _FLUSH_INTERVAL_SECONDS Local audit log file writes are flushed to filesystem at regular flush interval	60 (default)	Y
XAAUDIT.HDFS.LOCAL_BUFFER _ROLLOVER_INTERVAL_SECONDS Local audit log file is rotated to write to a new file at a rollover interval specified here	600 (default)	Y
XAAUDIT.HDFS.LOCAL_ARCHIVE _MAX_FILE_COUNT The maximum number of local audit log files will be kept in the archive directory	10 (default)	Y
SSL Information (https connectivity to policy Admin Tool)		

Configuration Property Name	Default/Example Value	Mandatory?
SSL_KEYSTORE_FILE_PATH Java Keystore Path where SSL key for the plug-in is stored. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	/etc/storm/conf/ranger-plugin-keystore.jks (default)	If SSL is enabled
SSL_KEYSTORE_PASSWORD Password associated with SSL Keystore. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	myKeyFilePassword (default)	If SSL is enabled
SSL_TRUSTSTORE_FILE_PATH Java Keystore Path where the trusted certificates are stored for verifying SSL connection to Policy Admin Tool. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	/etc/storm/conf/ranger-plugin-truststore.jks (default)	If SSL is enabled
SSL_TRUSTSTORE_PASSWORD Password associated with Truststore file. Is used only if SSL is enabled between Policy Admin Tool and Plugin; If SSL is not Enabled, leave the default value as it is - do not set as EMPTY if SSL not used.	changeit (default)	If SSL is enabled

3. Enable the Storm plug-in by entering the following commands:

```
cd /usr/hdp/<version>/ranger-storm-plugin
```

```
./enable-storm-plugin.sh
```

4. Restart Storm.

5. To confirm that the Storm plug-in installation and configuration are complete, go to the Audit Tab of the Ranger Admin Console and check Plugins. You should see Storm listed there.

13.5. Enabling Audit Logging for HDFS and Solr

The Ranger service provides the capability for you to enable audit logging for HDFS and/or Solr databases, which can be very helpful to maintain/query audit data when data grows to a significant amount.

To enable auditing for HDFS, perform the steps listed below.

1. Set the XAAUDIT.HDFS.ENABLE value to "true" for the component plug-in in the install.properties file, which can be found here:

```
/usr/hdp/<version>/ranger-<component>=plugin
```

2. Configure the NameNode host in the `XAAUDIT.HDFS.HDFS_DIR` field.

3. Create a policy in the HDFS service from the Ranger Admin for individual component users (`hive/hbase/knox/storm/yarn/kafka/kms`) to provide READ and WRITE permissions for the audit folder (i.e., for enabling Hive component to log Audits to HDFS, you need to create a policy for the hive user with Read and WRITE permissions for the audit directory).

4. Set the Audit to HDFS caches logs in the local directory, which can be specified in XAAUDIT.HDFS.LOCAL_BUFFER_DIRECTORY (this can be like `/var/log/<component>/**`), which is the path where the audit is stored for a short time. This is similar for archive logs that need to be updated.

To enable auditing reporting from the Solr database, perform the steps listed below.

1. Modify the following properties in the Ranger service `install.properties` to enable auditing to the Solr database in Ranger:

   - `audit_store=solr`
   - `audit_solr_urls=http://solr_host:6083/solr/ranger_audits`
   - `audit_solr_user=ranger_solr`
   - `audit_solr_password-NONE`

2. Restart Ranger.

To enable auditing to the Solr database for a plug-in (e.g., HBase), perform the steps listed below.

1. Set the following properties in `install.properties` of the plug-in to begin audit logging to the Solr database:

   - XAAUDIT.SOLR.IS.ENABLED=true
   - XAAUDIT.SOLR.ENABLE=true
   - XAAUDIT.SOLR.URL=http://solr_host:6083/solr/ranger_audits
   - XAAUDIT.SOLR.USER-ranger_solr
   - XAAUDIT.SOLR.PASSWORD=NONE
   - XAAUDIT.SOLR.FILE_SPOOL_DIR=/var/log/hadoop/hdfs/audit/solr/spool

2. Enable the Ranger HBase plug-in.

3. Restart the HBase component.

13.6. Verifying the Installation

To verify that installation was successful, perform the following checks:

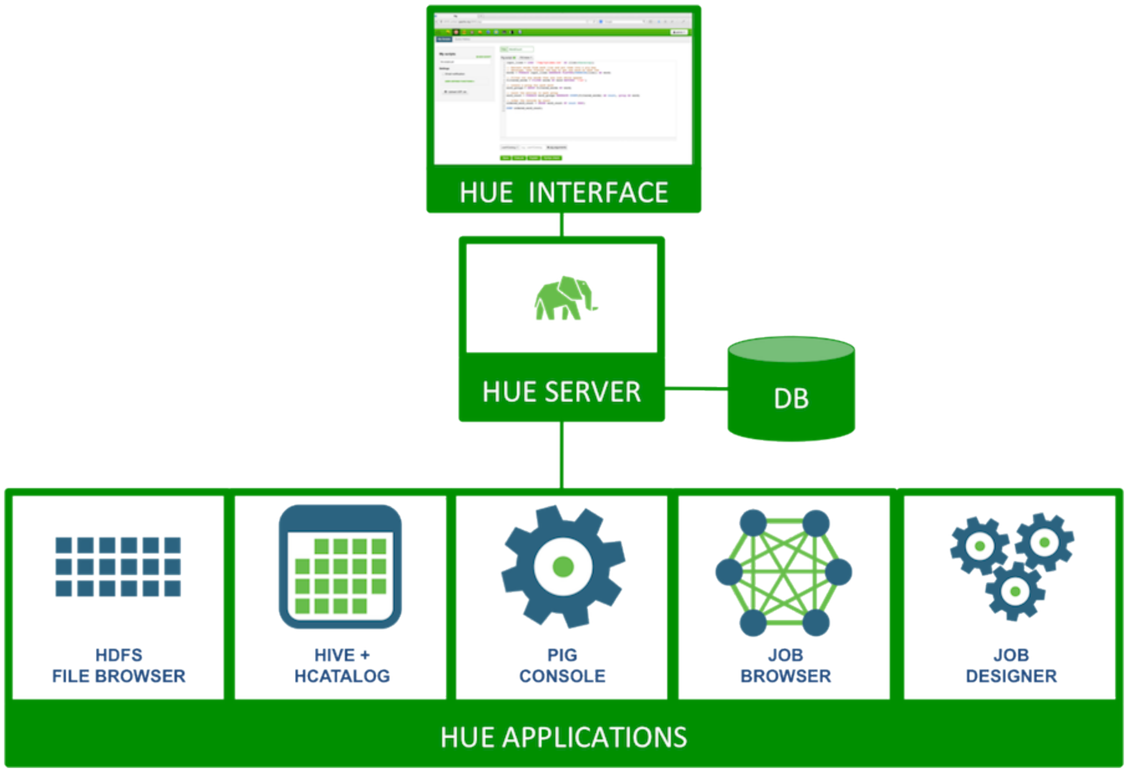
- Check whether the Database `RANGER_ADMIN_DB_NAME` is present in the MySQL server running on `RANGER_ADMIN_DB_HOST`

- Check whether the Database `RANGER_AUDIT_DB_NAME` is present in the MySQL server running on `RANGER_AUDIT_DB_HOST`
- Check whether the "ranger-admin" service is installed in services.msc (Windows only)
- Check whether the "ranger-usersync" service is installed in services.msc (Windows only)
- If you plan to use the Ranger Administration Console with the UserSync feature, check whether both services start
- Go to the Ranger administration console host URL and make sure you can log in using the default user credentials

14. Installing Hue

This chapter describes how to install, start, configure, and validat Hue.

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



14.1. Prerequisites

Note

Hue is not supported on Ubuntu or Debian.

Complete the following prerequisites before deploying Hue.

1. Verify that you have a host that supports Hue:

   Note

   Hue is not supported on CentOS 7 in HDP 2.3.

   - 64-bit Red Hat Enterprise Linux (RHEL) 6
   - 64-bit CentOS 6 • 64-bit Oracle Linux 6
   - 64-bit SUSE Linux Enterprise Server (SLES) 11 SP3/SP4

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

   Table 14.1. Hue-Supported Browsers

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. Verify that you have at least Python 2.6.6 or higher installed.

4. Stop all of the services in your cluster. For more information see the instructions provided in the Stopping HDP Services in the HDP Reference Guide.

5. Install and run the HDP Hadoop cluster from HDP-2.3.0.0.

   The following table outlines dependencies on HDP components:

   Table 14.2. Hue Dependencies on HDP Components

Component	Required	Applications	Notes
HDFS	Yes	Core, Filebrowser	HDFS access through WebHDFS or HttpFS
YARN	Yes	Core, Filebrowser	Transitive dependency via Hive or Oozie
Oozie	No	JobDesigner, Oozie	Oozie access through REST API
Hive	No	Hive, HCatalog	Beeswax server uses the Hive client libraries
WebHCat	No	HCatalog, Pig	HCatalog and Pig use WebHCat REST API

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

   You can deploy Hue on any host within your cluster. If your corporate firewall policies allow, you can also use a remote host machine as your Hue server. For evaluation or small cluster sizes, use the master install machine for HDP as your Hue server.

7. Configure the firewall.

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

- 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.

14.2. Configure HDP

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.

```
su - $HDFS_USER
```

```
/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop namenode
```

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

   On the NameNode, Secondary NameNode, and all the DataNodes, add the following property 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 `$WEBHCAT_CONF_DIR/webhcat-site.xml` file, where `$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` file, 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. Modify the `hive-site.xml` file. On the HiveServer2 host, add the following properties to the `$HIVE_CONF_DIR/hive-site.xml` file, where $HIVE_CONF_DIR is the directory for storing Hive configuration files. For example, `/etc/hive/conf`.

   ```
   vi $HIVE_CONF_DIR/hive-site.xml
   ```

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

14.3. Install Hue

Prerequisites

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

2. Verify the HDP repositories are available:

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

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

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

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

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

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

Installation

Run the following command on all Hue Server host machines:

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install hue
  ```

- For SLES:

  ```
  zypper install hue
  ```

Note

Not supported for Ubuntu or Debian.

14.4. 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
  ```

- As per the instructions below, modify the default configuration file provided: `/etc/hue/conf/hue.ini`.

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, 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. (Optional) 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:

   - Run the following command on the Hue Server:

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

   - 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

     ssl_cipher_list="DEFAULT:!aNULL:!eNULL:!LOW:!EXPORT:!
     SSLv2" (default)
     ```

     Note

     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
     ```

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

Configure Hadoop

Use the following instructions to configure Hadoop.

The following configuration variables are in the [hadoop] section of the `hue.ini` configuration file.

1. Configure an HDFS Cluster.

   Hue currently supports only one HDFS cluster. Ensure that you define the HDFS cluster under the [hadoop][[hdfs_clusters]] [[[default]]] subsection. Use the following variables to configure the HDFS cluster:

Table 14.3. Variables to Configure HDFS Cluster

Variable	Description	Default/Example Value
fs_defaultfs	This is equivalent to fs.defaultFS (fs.default.name) in the Hadoop configuration.	hdfs:// fqdn.namenode.host:8020
webhdfs_url	WebHDFS URL.	The default value is the HTTP port on the NameNode. Example: http://fqdn.namenode.host:50070/webhdfs/v1

2. Configure a 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:

Table 14.4. Variables to Configure the YARN Cluster

Variable	Description	Default/Example Value
submit_to	Set this property to true. Hue will submit jobs to this YARN cluster. Note that JobBrowser will not be able to show MR2 jobs.	true
resourcemanager_api_url	The URL of the ResourceManager API.	http://fqdn.resourcemanager.host:8088
proxy_api_url	The URL of the ProxyServer API.	http://fqdn.resourcemanager.host:8088
history_server_api_url	The URL of the HistoryServer API.	http://fqdn.historyserver.host:19888
node_manager_api_url	The URL of the NodeManager API.	http://fqdn.resourcemanager.host:8042

Note

For more information on how to configure Hue with a NameNode HA cluster, see "Deploy Hue with an HA Cluster."

Configure Beeswax

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

Table 14.5. Beeswax Configuration Values

Variable	Description	Default/Example Value
hive_server_host	Host where Hive server Thrift daemon is running. If Kerberos security is	

Variable	Description	Default/Example Value
	enabled, use fully-qualified domain name (FQDN).	
hive_server_port	Port on which HiveServer2 Thrift server runs.	10000
hive_conf_dir	Hive configuration directory where hive-site.xml is located.	/etc/hive/conf
server_conn_timeout	Timeout in seconds for Thrift calls to HiveServer2.	120

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.

Configure Hue to Communicate with HiveServer2 over SSL (Optional)

(Optional) Use the following changes to `hue.ini` to configure Hue to communicate with HiveServer2 over SSL:

```
[[ssl]]
SSL communication enabled for this server.
enabled=false
Path to Certificate Authority certificates.
cacerts=/etc/hue/cacerts.pem
Path to the public certificate file.
cert=/etc/hue/cert.pem
Choose whether Hue should validate certificates received from the server.
validate=true
```

Configure JobDesigner and Oozie

In the [liboozie] section of the configuration file, specify `oozie_url`, the URL of the Oozie service as specified by the OOZIE_URL environment variable for Oozie.

Configure WebHCat

In the [hcatalog] section of the `hue.ini` configuration file, set `templeton_url`, to the hostname or IP of the WebHCat server. For example: http:// hostname:50111/templeton/ v1/.

14.5. Start Hue

As a root user, start subprocesses corresponding to the different Hue components. Run the following command on the Hue Server: `/etc/init.d/hue start`

To stop Hue, run the following command: `/etc/init.d/hue stop`

To restart Hue, run the following command: `/etc/init.d/hue restart`

Checking the 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**.

14.6. Configuring Hue for an External Database

By default, Hue uses an embedded database, SQLite 3.6, but you can configure Hue to use any of the following external databases:

- Using Hue with Oracle [142]
- Using Hue with MySQL [143]
- Using Hue with PostgreSQL [144]

14.6.1. Using Hue with Oracle

To set up Hue to use an Oracle database:

1. Create a new user in Oracle and grant privileges to manage this database using the Oracle database admin utility:

   ```
   # sqlplus sys/root as sysdba
   CREATE USER $HUEUSER IDENTIFIED BY $HUEPASSWORD default tablespace
    “USERS”temporary tablespace “TEMP”;
   GRANT CREATE TABLE, CREATE SEQUENCE, CREATE PROCEDURE, CREATE TRIGGER,
    CREATE SESSION,
   UNLIMITED TABLESPACE TO $HUEUSER;
   ```

   Where $HUEUSER is the Hue user name and $HUEPASSWORD is the Hue user password.

2. Open the `/etc/hue/conf/hue.ini` file and edit the [[database]] section (modify for your Oracle setup).

   ```
   [[database]]
   engine=oracle
   host=$DATABASEIPADDRESSORHOSTNAME
   port=$PORT
   user=$HUEUSER
   password=$HUEPASSWORD
   name=$DBNAME
   ```

3. Install the Oracle instant clients and configure cx_Oracle.

   a. Download and extract the instantclient-basic-linux and instantclient-sdk-linux Oracle clients from Oracle's download site.

   b. Set your ORACLE_HOME environment variable to point to the newly downloaded client libraries.

   c. Set your LD_LIBRARY_PATH environment variable to include ORACLE_HOME.

   d. Create symbolic link for library expected by cx_Oracle:

```
ln -sf libclntsh.so.11.1 libclntsh.so
```

e. Install the cx_Oracle python module. Confirm that python-setuptools are present on Hue node, for example, `rpm -qa | grep python-setuptools`.

   If the python-setuptools are not present, install them, using the following command:

   ```
   yum install python-setuptools
   ```

f. Install the cx_Oracle module:

   ```
   /usr/lib/hue/build/env/bin/pip install cx_Oracle
   ```

g. Upgrade Django south:

   ```
   /usr/lib/hue/build/env/bin/pip install south --upgrade
   ```

4. Synchronize Hue with the external database to create the schema and load the data:

   ```
   /usr/lib/hue/build/env/bin/hue syncdb --noinput
   ```

   ```
   /usr/lib/hue/build/env/bin/hue migrate
   ```

5. Populate /usr/lib64 with Oracle instant-client library files.

   Copy the *.so.* files from oracle instantclient directory path to `/usr/lib64`.

6. Start Hue.

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

14.6.2. Using Hue with MySQL

To set up Hue to use a MySQL database:

1. Create a new user in MySQL, and grant privileges to it to manage the database using the MySQL database admin utility:

   ```
   # mysql -u root -p<
   CREATE USER $HUEUSER IDENTIFIED BY '$HUEPASSWORD';
   GRANT ALL PRIVILEGES on *.* to ‘$HUEUSER’@’localhost’ WITH GRANT OPTION;
   GRANT ALL on $HUEUSER.* to ‘$HUEUSER’@’localhost’ IDENTIFIED BY
    $HUEPASSWORD;
   FLUSH PRIVILEGES;
   ```

   where $HUEUSER is the Hue user name and $HUEPASSWORD is the Hue user password.

2. Create the MySQL database for Hue:

   ```
   # mysql -u root -p
   ```

   ```
   CREATE DATABASE $DBNAME;
   ```

3. Open the `/etc/hue/conf/hue.ini` file and edit the [[database]] section (modify for your MySQL setup).

```
[[database]]
engine=mysql
host=$DATABASEIPADDRESSORHOSTNAME
port=$PORT
user=$HUEUSER
password=$HUEPASSWORD
name=$DBNAME
```

4. Synchronize Hue with the external database to create the schema and load the data.

   ```
   /usr/lib/hue/build/env/bin/hue syncdb --noinput
   ```

5. Start Hue.

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

14.6.3. Using Hue with PostgreSQL

To set up Hue to use a PostgreSQL database:

1. Create a database in PostgreSQL using the PostgreSQL database admin utility.

   ```
   sudo -u postgres psql
   ```

   ```
   CREATE DATABASE $DBNAME;
   ```

2. Exit the database admin utility.

   ```
   \q <enter>
   ```

3. Create the Hue user.

   ```
   sudo -u postgres psql -d $DBNAME
   ```

   ```
   CREATE USER $HUEUSER WITH PASSWORD '$HUEPASSWORD';
   ```

   where $HUEUSER is the Hue user name and $HUEPASSWORD is the Hue user password.

4. Open the `/etc/hue/conf/hue.ini` file and edit the [[database]] section (modify for your PostgreSQL setup).

   ```
   [[database]]
   engine=postgresql_psycopg2
   host=$DATABASEIPADDRESSORHOSTNAME
   port=$PORT
   user=$HUEUSER
   password=$HUEPASSWORD
   name=$DBNAME
   ```

5. Install the PostgreSQL database adapter for Python (psycopg2). For RHEL/CentOS/Oracle Linux:

   ```
   yum install python-devel -y
   yum install postgresql-devel -y
   cd /usr/lib/hue
   source build/env/bin/activate
   ```

```
pip install psycopg2
```

6. Synchronize Hue with the external database to create the schema and load the data:

   ```
   /usr/lib/hue/build/env/bin/hue syncdb --noinput
   ```

7. Start Hue:

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

15. 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:

15.1. Install the Sqoop Package

Prerequisites

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

   ```
   yum list sqoop
   ```

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

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

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

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

Installation

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

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install sqoop
  ```

- For SLES:

  ```
  zypper install sqoop
  ```

15.2. Set Up the Sqoop Configuration

This section describes how to set up and edit the deployment configuration files for Sqoop. Hortonworks provides a set of configuration files that represent a working Sqoop configuration. (See Download Companion Files. You can use these files as a reference point. However, you will need to modify them to match your own cluster environment. If you choose to use the provided configuration files to set up your Sqoop environment, complete the following steps to set up the Sqoop configuration files:

1. 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.
2. Modify the configuration files.

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

   Also in `sqoop-env.sh`, make the following changes:

   - `export HADOOP_HOME=${HADOOP_HOME:-/usr/hdp/current/hadoop-client}`
   - `export HBASE_HOME=${HBASE_HOME:-/usr/hdp/current/hbase-client}`
   - `export HIVE_HOME=${{HIVE_HOME:-/usr/hdp/current/hive-server2}`
   - `export ZOOCFGDIR=${ZOOCFGDIR:-/etc/zookeeper/conf}`
   - Copy all the configuration files to the Sqoop configuration directory, such as `/etc/sqoop/conf`.
   - Add the following entry to `/usr/bin/sqoop`:

     `export HIVE_HOME=$ {HIVE_HOME:-/usr/hdp/<version>/hive-server2}` where <version> is the same HDP version as the other entries in `/usr/bin/sqoop`

15.3. Validate the Sqoop Installation

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

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

16. Installing Apache Mahout

Install Apache Mahout on the server on which it will run, either the Hadoop master node or your client host. Because Mahout consists of client software, there is no need to install it on every node in your cluster.

16.1. Install Mahout

To install the Mahout package, use the following command:

- RHEL/CentOS/Oracle Linux:

  ```
  yum install mahout
  ```

- For SLES:

  ```
  zypper install mahout
  ```

16.2. Validate Mahout

To validate Mahout:

1. Create a test user named "testuser" on the client host, the Linux cluster, and in HDFS, then log in to the client host as user.

   ```
   hdfs dfs -put /tmp/sample-test.txt /user/testuser
   ```

2. Export the required environment variables for Mahout:

   ```
   export JAVA_HOME=<your jdk home install location here>
   export HADOOP_HOME=/usr/hdp/current/hadoop-client
   export MAHOUT_HOME=/usr/hdp/current/mahout-client
   export PATH="$PATH":$HADOOP_HOME/bin:$MAHOUT_HOME/bin
   export CLASSPATH="$CLASSPATH":$MAHOUT_HOME
   ```

3. Upload a few megabytes of natural-language plain text to the client host as `/tmp/sample-test.txt`.

4. Transfer the `sample-test.txt` file to a subdirectory of the testuser's HDFS home directory.

   ```
   hdfs dfs -mkdir /user/testuser/testdata
   hdfs dfs -put /tmp/sample-test.txt /user/testuser/testdata
   ```

5. Create a mahout test output directory:

   ```
   hdfs dfs -mkdir /user/testuser/mahouttest
   ```

6. Use the following command to instruct 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/testdata --output /
user/ testuser/mahouttest -ow --charset utf-8
```

17. Installing and Configuring Apache Flume

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:

17.1. Understanding 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/hdp/current/flume-server/docs/index.html` on the host on which Flume is installed.

The “Flume User Guide” is available at `file:////usr/hdp/current/flume-server/docs/FlumeUserGuide.html`. If you have access to the Internet, the same documentation is also available at the Flume website.

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 to 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. An agent must contain at least one source, channel, and sink, but the same agent can run multiple sources, sinks, and channels. A particular data flow path is set up through the configuration process.

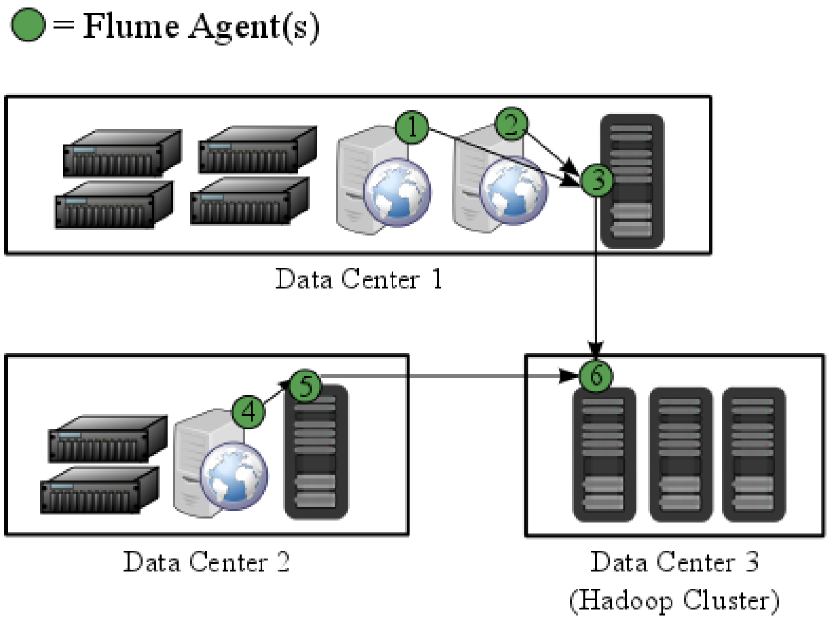
17.2. Installing Flume

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 Configuring Flume for more information about configuring Flume agents.

Prerequisites

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

   ```
   yum list flume
   ```

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

   ```
   flume.noarch 1.5.2.2.2.6.0-2800.el6 HDP-2.3
   ```

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

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

3. You must have set up your JAVA_HOME environment variable per your operating system. See JDK Requirements for instructions on installing JDK.

   ```
   export JAVA_HOME=/path/to/java
   ```

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

   Table 17.1. Flume 1.5.2 Dependencies

Flume Component	HDP Component Dependencies
HDFS Sink	Hadoop 2.4
HBase Sink	HBase 0.98.0
Hive Sink	Hive 0.13.0, HCatalog 0.13.0, and Hadoop 2.4

Installation

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

To install Flume from a terminal window, type:

- For RHEL or CentOS:

  ```
  yum install flume
  ```

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

- For SLES:

```
zypper install flume
```

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

The main Flume files are located in `/usr/hdp/current/flume-server`. The main configuration files are located in `/etc/flume/conf`.

17.3. Configuring Flume

To configure a Flume agent, edit the following three configuration files:

- flume.conf
- flume-env.sh
- log4j.properties

flume.conf

Configure each Flume agent by defining properties in a configuration file at `/etc/flume/conf/flume.conf`. The init scripts installed by the flume-agent package read the contents of this file when starting a Flume agent on any host. At a minimum, the Flume configuration file must specify the required sources, channels, and sinks for your Flume topology.

For example, the following sample Flume configuration file defines a Netcat source, a Memory channel and a Logger sink. This configuration lets a user generate events and subsequently logs them to the console.

```
# example.conf: A single-node Flume configuration

# Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

# Describe/configure the source
a1.sources.r1.type = netcat
a1.sources.r1.bind = localhost
a1.sources.r1.port = 44444

# Describe the sink
a1.sinks.k1.type = logger

# Use a channel that buffers events in memory
a1.channels.c1.type = memory
a1.channels.c1.capacity = 1000
a1.channels.c1.transactionCapacity = 100

# Bind the source and sink to the channel
a1.sources.r1.channels = c1
```

```
a1.sinks.k1.channel = c1
```

This configuration defines a single agent named a1. a1 has a source that listens for data on port 44444, a channel that buffers event data in memory, and a sink that logs event data to the console. The configuration file names the various components, and describes their types and configuration parameters. A given configuration file might define several named agents.

See the Apache Flume 1.5.2 User Guide for a complete list of all available Flume components.

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

A second template file exists for setting environment variables automatically at startup:

`/etc/flume/conf/flume- env.sh.template`.

Note

If you use an HDFS sink, be sure to specify a target folder in HDFS.

flume-env.sh

Set environment options for a Flume agent in `/etc/flume/conf/flume-env.sh`:

- To enable JMX monitoring, add the following properties to the JAVA_OPTS property:

```
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 customize the heap size, add the following properties to the JAVA_OPTS property:

```
JAVA_OPTS= "-Xms100m -Xmx4000m"
```

log4j.properties

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

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

17.4. Starting Flume

There are two options for starting Flume.

- To start Flume directly, run the following command on the Flume host:

```
/usr/hdp/current/flume-server/bin/flume-ng agent -c /etc/flume/
conf -f /etc/flume/conf/ flume.conf -n agent
```

- To start Flume as a service, run the following command on the Flume host:

```
service flume-agent start
```

17.5. HDP and Flume

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

Sources

- Exec (basic, restart)
- Syslogtcp
- Syslogudp

Channels

- Memory
- File

Sinks

- HDFS: secure, nonsecure
- HBase

See the Apache Flume 1.5.2 User Guide for a complete list of all available Flume components.

17.6. A Simple Example

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

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory

agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd

agent.sinks = hdfsSinkagent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://hdp/user/root/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

The source here is defined as an exec source. The agent runs a given command on startup, 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.

18. 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.

To install Apache Storm, complete the following instructions.

Note

To install and configure Storm on an Ambari-managed cluster, refer to Adding a Service in the *Ambari User's Guide*.

To configure Storm for Kerberos in an Ambari-Managed Cluster, refer to Configuring Storm for Kerberos in an Ambari-Managed Cluster.

18.1. Install the Storm Package

Prerequisite: Storm requires version 2.6 or higher of the default system Python interpreter.

1. To install the Storm RPMs, run the following command on each client cluster node and gateway node:

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install storm
     ```

   - For SLES:

     ```
     zypper install storm
     ```

2. Run the following command to create the conf directory:

   ```
   sudo mkdir -p /etc/storm/conf
   ```

3. Run the following command to create the `storm.yaml` file:

   ```
   sudo touch /etc/storm/conf/storm.yaml
   ```

18.2. Configure Storm

Use the following procedure to configure Storm:

1. Add the following properties to the `/etc/storm/conf/storm.yaml` file, substituting your own list of hostnames and ports:

   ```
   storm.zookeeper.servers: [<zookeeper-servers>]
   nimbus.seeds: [<nimbus-hostnames>]
   storm.local.dir: $STORM_LOCAL_DIR
   logviewer.port: 8081
   ```

   where:

   <zookeeper-servers> is a comma-separated list of ZooKeeper servers.

   <nimbus-hostnames> is a comma-separated list of hosts where the Storm Nimbus server is started.

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

   For example:

   ```
   storm.zookeeper.servers: ["host1:port1", "host2:port2", "host3:port3"]
   nimbus.seeds: ["host1:port1", "host2:port2"]
   storm.local.dir: /mnt/storm
   logviewer.port: 8081
   ```

2. Run the following commands:

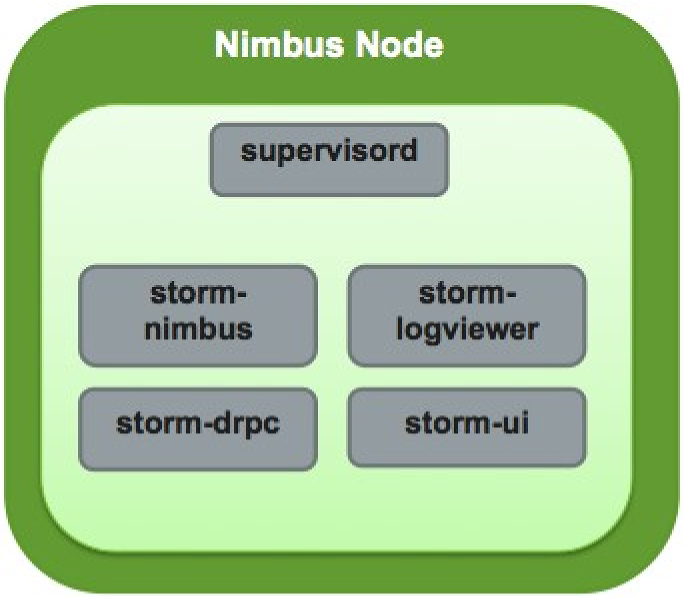
   ```
   chown -R storm:storm $STORM_LOCAL_DIR
   ```

   ```
   chmod -R 755 $STORM_LOCAL_DIR
   ```

18.3. Configure a 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 administrators may use another process controller of their choice, such as monitor daemontools.



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

18.4. (Optional) Configure Kerberos Authentication for Storm

Storm supports authentication using several models. This topic describes how to configure your Storm installation to use Kerberos authentication. At a high level, administrators must perform the tasks in this section.

Create Keytabs and Principals for Storm Daemons

Storm requires a principal and keytab when using Kerberos for authentication. A principal name in a given realm consists of a primary name and an instance name, the FQDN of the host that runs the service, in this case Storm. As services do not log in with a password to acquire their tickets, the authentication credentials for the service principal are stored in a keytab file, which is extracted from the Kerberos database and stored locally with the service principal on the service component host. First, create the principal using mandatory naming conventions. Then, create the keytab file with information from the new principal and copy the keytab to the keytab directory on the appropriate Storm host.

Note

Principals can be created either on the Kerberos Key Distribution Center (KDC) host or over the network using an “admin” principal. The following instructions assume you are using the KDC machine and using the kadmin.local command line administration utility. Using kadmin.local on the KDC machine allows you to create principals without needing to create a separate "admin" principal before you start.

Perform the following procedure on the host that runs KDC:

1. Make sure that you have performed the steps in Securing ZooKeeper with Kerberos.
2. Create a principal for the Nimbus server and the Storm DRPC daemon:

   ```
   sudo kadmin.local -q 'addprinc storm/
   <STORM_HOSTNAME>@STORM.EXAMPLE.COM'
   ```

3. Create a keytab for the Nimbus server and the Storm DRPC daemon:

   ```
   sudo kadmin.local -q "ktadd -k /tmp/storm.keytab storm/
   <STORM_HOSTNAME>@STORM.EXAMPLE.COM"
   ```

4. Copy the keytab to the Nimbus node and the node that runs the Storm DRPC daemon.
5. Run the following command to create a principal for the Storm UI daemon, the Storm Logviewer daemon, and the nodes running the process controller, such as Supervisor. A process controller is used to start and stop the Storm daemons.

   ```
   sudo kadmin.local -q 'addprinc storm@STORM.EXAMPLE.COM'
   ```

6. Create a keytab for the Storm UI daemon, the Storm Logviewer daemon, and Supervisor:

```
sudo kadmin.local -q "ktadd -k /tmp/storm.keytab
storm@STORM.EXAMPLE.COM"
```

7. Copy the keytab to the cluster nodes running the Storm UI daemon, the Storm Logviewer daemon, and Supervisor.

Update the jaas.conf Configuration File

Both Storm and ZooKeeper use Java Authentication and Authorization Services (JAAS), an implementation of the Pluggable Authentication Model (PAM), to authenticate users. Administrators must update the `jaas.conf` configuration file with the keytab and principal information from the last step. The file must appear on all Storm nodes, the Nimbus node, the Storm DRPC node, and all Gateway nodes. However, different cluster nodes require different stanzas, as indicated in the following table:

Table 18.1. Required jaas.conf Sections for Cluster Nodes

Cluster Node	Required Sections in jaas.conf
Storm	StormClient
Nimbus	StormServer, Client
DRPC	StormServer
Supervisor	StormClient, Client
Gateway	StormClient (different structure than used on Storm and Supervisor nodes)
ZooKeeper	Server

Note

JAAS ignores unnecessary sections in `jaas.conf`. Administrators can put all sections in all copies of the file to simplify the process of updating it. However, the StormClient stanza for the Gateway nodes uses a different structure than the StormClient stanza on other cluster nodes. In addition, the StormServer stanza for the Nimbus node requires additional lines, as does the zoo.cfg configuration file for the ZooKeeper nodes.

The following example `jaas.conf` file contains all sections and includes information about the keytabs and principals generated in the previous step.

```
StormServer {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/keytabs/storm.keytab"
storeKey=true
useTicketCache=false
principal="storm/storm.example.com@STORM.EXAMPLE.COM";
};

StormClient {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/keytabs/storm.keytab"
storeKey=true
useTicketCache=false
serviceName="storm"
principal="storm@STORM.EXAMPLE.COM";
};

Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/keytabs/storm.keytab"
storeKey=true
useTicketCache=false
serviceName="zookeeper"
principal="storm@STORM.EXAMPLE.COM";
};
```

The StormServer section for the Nimbus node must have the following additional lines:

```
StormServer {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/keytabs/storm.keytab"
storeKey=true
useTicketCache=false
principal="storm/storm.example.com@STORM.EXAMPLE.COM";
};
```

The StormClient stanza for the Gateway nodes must have the following structure:

```
StormClient {
com.sun.security.auth.module.Krb5LoginModule required
doNotPrompt=false
useTicketCache=true
serviceName="$nimbus_user";
};
```

The Server stanza for the ZooKeeper nodes must have the following structure:

```
Server {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
keyTab="/keytabs/zk.keytab"
storeKey=true
useTicketCache=false
serviceName="zookeeper"
principal="zookeeper/zk1.example.com@STORM.EXAMPLE.COM";
};
```

In addition, add the following `childopts` lines to the stanzas for the nimbus, ui, and supervisor processes:

```
nimbus.childopts: "-Xmx1024m -Djava.security.auth.login.config=/path/to/jaas.
conf"
ui.childopts: "-Xmx768m -Djava.security.auth.login.config=/path/to/jaas.conf"
supervisor.childopts: "-Xmx256m -Djava.security.auth.login.config=/path/to/
jaas.conf"
```

Note

When starting ZooKeeper, include the following command-line option so that ZooKeeper can find jaas.conf:

```
        -Djava.security.auth.login.config=/jaas/zk_jaas.conf
```

Update the storm.yaml Configuration File

To enable authentication with Kerberos, add the following lines to the `storm.yaml` configuration file:

```
storm.thrift.transport: "backtype.storm.security.auth.kerberos.
KerberosSaslTransportPlugin"
java.security.auth.login.config: "/path/to/jaas.conf"
nimbus.authorizer: "backtype.storm.security.auth.authorizer.
SimpleACLAuthorizer"
storm.principal.tolocal: "backtype.storm.security.auth.
KerberosPrincipalToLocal"
storm.zookeeper.superACL: "sasl:storm"
nimbus.admins: - "storm"
nimbus.supervisor.users: - "storm"
nimbus.childopts: "-Xmx1024m -Djavax.net.debug=ssl -Dsun.security.krb5.
debug=true -Djava.security.auth.login.config=/vagrant/storm_jaas.conf -Djava.
security.krb5.realm=EXAMPLE.COM -Djava.security.krb5.kdc=kdc.example.com"
ui.childopts: "-Xmx768m -Djavax.net.debug=ssl -Dsun.security.krb5.debug=true
 -Djava.security.auth.login.config=/vagrant/storm_jaas.conf -Djava.security.
krb5.realm=EXAMPLE.COM -Djava.security.krb5.kdc=kdc.example.com"
supervisor.childopts: "-Xmx256m -Djavax.net.debug=ssl -Dsun.security.krb5.
debug=true -Djava.security.auth.login.config=/vagrant/storm_jaas.conf -Djava.
security.krb5.realm=EXAMPLE.COM -Djava.security.krb5.kdc=example.host1.com"
ui.filter: "org.apache.hadoop.security.authentication.server.
AuthenticationFilter"
ui.filter.params: "type": "kerberos""kerberos.principal": "HTTP/nimbus.
example.com""kerberos.keytab": "/vagrant/keytabs/http.keytab""kerberos.name.
rules": "RULE:[2:$1@$0]([jt]t@.*EXAMPLE.COM)s/.*/$MAPRED_USER/ RULE:[2:$1@$0]
([nd]n@.*EXAMPLE.COM)s/.*/$HDFS_USER/DEFAULT"
```

18.5. (Optional) Configuring Authorization for Storm

Apache Storm supports authorization using Pluggable Authentication Modules, or PAM, with secure Hadoop clusters. Currently, Storm supports the following authorizers:

Table 18.2. Supported Authorizers

Authorizer	Description
backtype.storm.security.auth.authorizer. SimpleACLAuthorizer	Default authorizer for the Nimbus node and all Storm nodes except DRPC.
backtype.storm.security.auth.authorizer. DRPCSimpleACLAuthorizer	Default authorizer for Storm DRPC nodes.
com.xasecure.authorization.storm.authorizer. XaSecureStormAuthorizer	Default authorizer for centralized authorization with Apache Ranger.

To enable authorization, perform the following steps:

1. Configure `storm.yaml` for Nimbus and Storm nodes.
2. Configure `worker-launcher.cfg` for worker-launcher.

3. Configure the Storm multi-tenant job scheduler.

Configure storm.yaml for Nimbus and Storm Nodes

When authorization is enabled, Storm prevents users from seeing topologies run by other users in the Storm UI. To do this, Storm must run each topology as the operating system user who submitted it rather than the user that runs Storm, typically storm, which is created during installation.

Use the following procedure to configure supervisor to run Storm topologies as the user who submits the topology, rather than as the storm user:

1. Verify that a headless user exists for supervisor, such as supervisor, on each Storm cluster node.
2. Create a headless operating system group, such as supervisor, on each Storm cluster node.
3. Set the following configuration properties in the storm.yaml configuration file for each node in the Storm cluster:

Table 18.3. storm.yaml Configuration File Properties

Configuration Property	Description
supervisor.run.worker.as.user	Set to true to run topologies as the user who submits them.
topology.auto-credentials	Set to a list of Java plugins that pack and unpack user credentials for Storm workers. This should be set to backtype.storm.security.auth.kerberos.AutoTGT.
drpc.authorizer	Set to backtype.storm.security.auth.authorizer. DRPCSimpleACLAuthorizer to enable authorizer for Storm DRPC node.
nimbus.authorizer:	Set to "backtype.storm.security.auth.authorizer.SimpleACLAuthorizer" to enable authorizer for Storm nimbus node.
storm.principal.tolocal:	Set to "backtype.storm.security.auth.KerberosPrincipalToLocal" enable transforming kerberos principal to local user names.
storm.zookeeper.superACL:	"sasl:storm" This will set the acls on zookeeper nodes so only user storm can modify those nodes.

4. Change the owner of `worker-launcher.cfg` to root and verify that only root has write permissions on the file.
5. Change the permissions for the worker-launcher executable to 6550.
6. Verify that all Hadoop configuration files are in the CLASSPATH for the Nimbus server.
7. Verify that the nimbus operating system user has superuser privileges and can receive delegation tokens on behalf of users submitting topologies.
8. Restart the Nimbus server.

Configure worker-launcher.cfg

`/usr/hdp/current/storm-client/bin/worker-launcher` is a program that runs Storm worker nodes. You must configure worker-launcher to run Storm worker nodes as the user who submitted a topology, rather than the user running the supervisor process controller. To do this, set the following configuration properties in the `/etc/storm/conf/worker-launcher.cfg` configuration file on all Storm nodes:

Table 18.4. worker-launcher.cfg File Configuration Properties

Configuration Property	Description
storm.worker-launcher.group	Set this to the headless OS group that you created earlier.
min.user.id	Set this to the first user ID on the cluster node.

Configure the Storm Multi-tenant Scheduler

The goal of the multi-tenant scheduler is to both isolate topologies from one another and to limit the resources that an individual user can use on the cluster. Add the following configuration property to `multitenant-scheduler.yaml` and place it in the same directory with `storm.yaml`.

Table 18.5. multitenant-scheduler.yaml Configuration File Properties

Configuration Property	Description
multitenant.scheduler.user.pools	Specifies the maximum number of nodes a user may use to run topologies.

The following example limits users evans and derek to ten nodes each for all their topologies:

```
multitenant.scheduler.user.pools: "evans": 10 "derek": 10
```

Note

The multi-tenant scheduler relies on Storm authentication to distinguish between individual Storm users. Verify that Storm authentication is already enabled.

18.6. Validate the Installation

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

Important

You must start ZooKeeper before starting Storm.

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

   - RHEL/CentOS/Oracle Linux

     ```
     etc/init.d/supervisor start
     ```

   - SLES

     ```
     etc/init.d/supervisor start
     ```

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

   - RHEL/CentOS/Oracle Linux

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

   - SLES

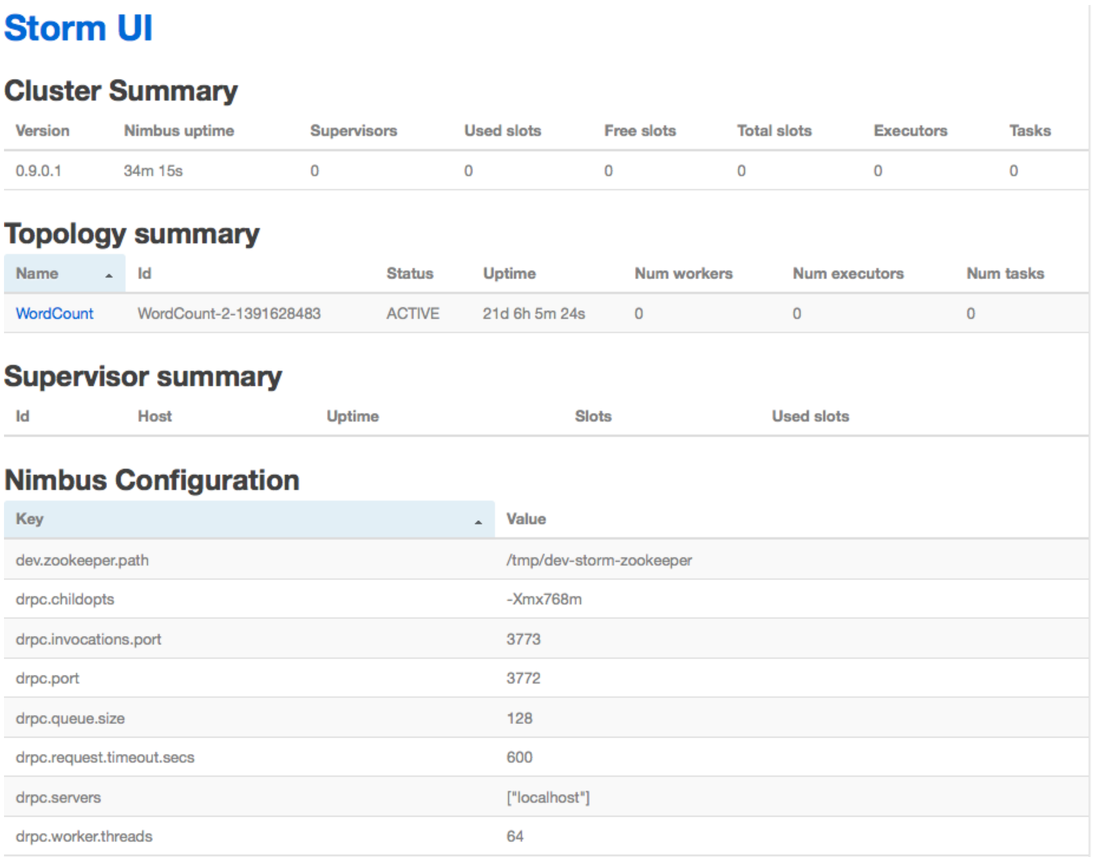
     ```
     /usr/bin/supervisorctl status
     ```

   ```
   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 the Storm UI web page:



Storm UI

Cluster Summary

Version	Nimbus uptime	Supervisors	Used slots	Free slots	Total slots	Executors	Tasks
0.9.0.1	34m 15s	0	0	0	0	0	0

Topology summary

Name	Id	Status	Uptime	Num workers	Num executors	Num tasks
WordCount	WordCount-2-1391628483	ACTIVE	21d 6h 5m 24s	0	0	0

Supervisor summary

Id	Host	Uptime	Slots	Used slots

Nimbus Configuration

Key	Value
dev.zookeeper.path	/tmp/dev-storm-zookeeper
drpc.childopts	-Xmx768m
drpc.invocations.port	3773
drpc.port	3772
drpc.queue.size	128
drpc.request.timeout.secs	600
drpc.servers	["localhost"]
drpc.worker.threads	64

4. Run the following command to run the WordCount sample topology:

   ```
   storm jar /usr/hdp/current/storm-client/contrib/storm-
   starter/storm-starter-*.*-jar-with-dependencies.jar
   storm.starter.WordCountTopology wordcount
   ```

19. Installing and Configuring Apache Spark

This section describes how to install and configure Apache Spark for HDP:

For more information about Spark on HDP (including how to install Spark using Ambari), see the Apache Spark Quick Start Guide.

19.1. Spark Prerequisites

Before installing Spark, make sure your cluster meets the following prerequisites:

Table 19.1. Spark Cluster Prerequisites

Item	Prerequisite
Cluster Stack Version	• HDP 2.2.6 or later
(Optional) Ambari Version	• 2.1 or later
Components	• Spark requires HDFS and YARN • PySpark requires Python to be installed on all nodes

Note

If you installed the Spark tech preview, save any configuration changes you made to the tech preview environment. Install Spark, and then update the configuration with your changes.

19.2. Installing Spark

When you install Spark, two directories will be created:

- `/usr/hdp/current/spark-client` for submitting Spark jobs
- `/usr/hdp/current/spark-history` for launching Spark master processes, such as the Spark history server

1. Search for Spark in the HDP repo:

   - For RHEL or CentOS:

     ```
     yum search spark
     ```

   - For SLES:

```
zypper install spark
```

- For Ubuntu and Debian:

```
apt-cache spark
```

This will show all the versions of Spark available. For example,

```
spark_2_3_2_0_2950-master.noarch : Server for Spark master
spark_2_3_2_0_2950-python.noarch : Python client for Spark
spark_2_3_2_0_2950-worker.noarch : Server for Spark worker
```

2. Install the version corresponding to the HDP version you currently have installed.

- For RHEL or CentOS:

```
yum install spark_<version>-master spark_<version>-python
```

- For SLES:

```
zypper install spark_<version>-master spark_<version>-python
```

- For Ubuntu and Debian:

```
apt-get install spark_<version>-master apt-get install
spark_<version>-python
```

19.3. Configuring Spark

To configure Spark, edit the following configuration files on all nodes that will run Spark jobs. These configuration files reside in the Spark client conf directory `/usr/hdp/current/spark-client/conf` on each node.

- `java-opts`
- If you plan to use Hive with Spark, `hive-site.xml`
- `spark-env.sh`
- `spark-defaults.conf`

Note

Note: the following instructions are for a non-Kerberized cluster.

java-opts

Create a `java-opts` file in the Spark client `/conf` directory. Add the following line to the file.

```
-Dhdp.version=<HDP-version>
```

For example:

```
-Dhdp.version=2.3.0.0-2800
```

hive-site.xml

If you plan to use Hive with Spark, create a hive-site.xml file in the Spark client /conf directory. (Note: if you installed the Spark tech preview you can skip this step.)

In this file, add the hive.metastore.uris property and specify the Hive metastore as its value:

```
<property>
     <name>hive.metastore.uris</name>
     <value>thrift://c6401.ambari.apache.org:9083</value>
</property>
```

spark-env.sh

Create a spark-env.sh file in the Spark client /conf directory, and make sure the file has the following entries:

```
# Location where log files are stored (default: ${SPARK_HOME}/logs)
# This can be any directory where the spark user has R/W access
export SPARK_LOG_DIR=/var/log/spark

# Location of the pid file (default: /tmp)
# This can be any directory where the spark user has R/W access
export SPARK_PID_DIR=/var/run/spark
```

These settings are required for starting Spark services (for example, the History Service and the Thrift server). The user who starts Spark services needs to have read and write permissions to the log file and PID directory. By default these files are in the `$SPARK_HOME` directory, typically owned by root in RMP installation.

We recommend that you set HADOOP_CONF_DIR to the appropriate directory; for example:

```
set HADOOP_CONF_DIR=/etc/hadoop/conf
```

This will minimize the amount of work you need to do to set up environment variables before running Spark applications.

spark-defaults.conf

Edit the `spark-defaults.conf` file in the Spark client `/conf` directory. Make sure the following values are specified, including hostname and port. (Note: if you installed the tech preview, these will already be in the file.) For example:

```
spark.yarn.historyServer.address c6401.ambari.apache.org:18080
spark.history.ui.port 18080
spark.yarn.services org.apache.spark.deploy.yarn.history.YarnHistoryService
spark.driver.extraJavaOptions -Dhdp.version=2.3.0.0-2800
spark.yarn.am.extraJavaOptions -Dhdp.version=2.3.0.0-2800
```

Create a Spark user

To use the Spark History Service, run Hive queries as the Spark user, or run Spark jobs; the associated user must have sufficient HDFS access. One way of ensuring this is to add the user to the hdfs group.

The following example creates a spark user:

- Create the Spark user on all nodes. Add it to the hdfs group.

  `useradd spark` This command is only required for tarball spark installs, not rpm-based installs.

  ```
  usermod -a -G hdfs spark
  ```

- Create the spark user directory under `/user/spark`:

  ```
  sudo su $HDFS_USER
  ```

  ```
  hdfs dfs -mkdir -p /user/spark
  ```

  ```
  hdfs dfs -chown spark:spark /user/spark
  ```

  ```
  hdfs dfs -chmod -R 755 /user/spark
  ```

19.4. Validating Spark

To validate the Spark installation, run the Spark jobs in the Validating Spark section of the Apache Spark Quick Start Guide.

20. Installing and Configuring Apache Kafka

This section describes how to install Apache Kafka, a high-throughput messaging system with publish-and-subscribe semantics. Kafka is often used in place of traditional message brokers like JMS and AMQP because of its higher throughput, replication, and fault tolerance.

To install Apache Kafka, complete the following instructions:

1. Install Kafka [170]
2. Configure Kafka [171]
3. Validate Kafka [172]

20.1. Install Kafka

Prerequisites and Considerations

When installing Kafka, note the following prerequisites and considerations:

- Administrators must use Apache ZooKeeper to manage Kafka for an HDP cluster. Verify that you have successfully installed ZooKeeper before installing and configuring Kafka.
- Kafka does not currently support user authentication and authorization.
- The following underlying file systems are supported for use with Kafka:
  - EXT3: recommended
  - EXT2
  - EXT4: supported, but might have performance issues (see the "EXT4 Notes" section in Apache production system notes.

Caution

Encrypted file systems such as SafenetFS are not supported for Kafka. Index file corruption can occur.

Installation

Install the Kafka RPMs or packages by completing the following steps.

Note

Hortonworks recommends avoiding using multiple brokers in a single node for Kafka. However, if you need to install a multi-node cluster, use the following instructions to install Kafka on another machine, make sure each `broker.id` is unique on the cluster, and ensure that `zookeeper.connect` is the same for all brokers.

1. Run the following command on each client cluster node and gateway node:

   - For RHEL/CentOS/Oracle Linux

     ```
     yum install kafka
     ```

   - For SLES

     ```
     zypper install kafka
     ```

2. Check the JAVA_HOME environment variable. If it has not yet been set, add the following to the PATH variable:

   ```
   export JAVA_HOME=<path of installed jdk version folder>
   ```

20.2. Configure Kafka

Use the following procedure to configure Kafka.

1. By default, Kafka is installed at `/usr/hdp/current/kafka-broker`.

2. Verify the values of the following configuration properties in the `server.properties` file:

Table 20.1. Kafka Configuration Properties

Kafka Configuration Property	**Description**
broker.id	Each Kafka broker requires a unique integer as an identifier. The default value is 0.
port	The port to which the Kafka socket server listens. The default value is 9092.
log.dirs	Comma-separated list of directories where Kafka log files are stored. The default value is /tmp/kafka-logs.
zookeeper.connect	The hostname or IP address of the host running ZooKeeper and the port to which ZooKeeper listens. The default value is localhost:2181.
log.retention.hours	The number of hours to wait before a Kafka log file is eligible for deletion. The default value is 168 hours (7 days).
Listeners	listener - Comma-separated list of URIs we will listen on and their protocols. Specify hostname as 0.0.0.0 to bind to all interfaces. Leave hostname empty to bind to default interface. Examples of legal listener lists: PLAINTEXT://myhost:9092,PLAINTEXTSASL://:9091 PLAINTEXT://0.0.0.0:9092, PLAINTEXTSASL://localhost:9093

20.3. Validate Kafka

Use the following procedure to verify the Kafka installation and configuration.

Note

Verify that ZooKeeper is running before starting Kafka and validating the installation.

1. Start the Kafka service using user kafka:

   ```
   su kafka -c "KAFKA_HOME/bin/kafka start"
   ```

2. Create a Kafka topic with the name "test" that has a replication factor of 1 and 1 partition.

   ```
   bin/kafka-topics.sh --zookeeper localhost:2181 --create --topic
   test --replication-factor 1 --partitions 1
   ```

   After running this command, you should see the following output:

   ```
   Created topic "test"
   ```

   Note

   The value of `--replication-factor` must be less then or equal to the number of Kafka brokers in the cluster. Otherwise an error will occur. Usually the replication-factor equals the number of Kafka brokers in the cluster.

3. Start a command line Kafka console producer that you can use to send messages. You can type your message once the process is started.

   ```
   <KAFKA_HOME>/bin/kafka-console-producer.sh --broker-list
   localhost:9092 --topic test
   ```

   You should see your test message, for example:

   ```
   This is a message.
   ```

   Note

   To return to the command prompt after sending the test message, type Ctrl + C.

4. Start a command line kafka consumer that you can use to read the messages sent by the producer.

   ```
   <KAFKA_HOME>/bin/kafka-console-consumer.sh --zookeeper
   localhost:2181 --topic test --from-beginning
   ```

21. Installing Apache Accumulo

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

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. We recommend that you run Network Time Protocol (NTP) within the cluster to keep node clocks in sync, to avoid problems with automatically timestamped data.

21.1. Installing the Accumulo Package

Prerequisites

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

2. Verify the HDP repositories are available:

   ```
   yum list accumulo
   ```

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

   ```
   accumulo.noarch <version>
   ```

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

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

Installation

To install the Accumulo RPM or package, use the following command:

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install accumulo
  ```

- For SLES:

```
zypper install accumulo
```

21.2. Configuring Accumulo

1. Accumulo provides example configurations that you can modify. Copy all files from one of the examples folders in `/etc/accumulo/conf/examples` to `/etc/accumulo/conf`.

   For example, you would use the following command to copy all files in the `/etc/accumulo/conf/examples/512MB/standalone` folder to the `/etc/accumulo/conf` folder:

   ```
   cp /etc/accumulo/conf/examples/512MB/standalone/* /etc/accumulo/
   conf
   ```

2. Accumulo has the option to use a native library that manages the memory used for newly written data outside of the Java heap for the Tablet Servers. This allows Accumulo to manage its memory more efficiently, improving performance. Use of the native library should be enabled whenever possible. To build the native library for your system, run the following on each host:

   ```
   JAVA_HOME=path_to_java_home /usr/hdp/current/accumulo-client/
   bin/build_native_library.sh
   ```

   Once this is done, various configuration properties must be changed to use the native maps, with examples appearing in the `/etc/accumulo/conf/examples/native-standalone` folder.

   ### Note

   If native maps are not enabled, the examples in the standalone folder should be used instead.

3. Make an Accumulo data directory:

   ```
   su - hdfs
   ```

   ```
   hadoop fs -mkdir -p /apps/accumulo
   ```

4. The example configuration files include an `accumulo-site.xml` file. Add the following property to this file to reference the Accumulo data directory:

   ### Note

   Change the value of the instance.secret in the `accumulo-site.xml` file, and then change the permissions on the file to 700 to protect the instance.secret from being readable by other users.

   ```
   <property>
        <name>instance.volumes</name>
        <value>hdfs://namenode:port/apps/accumulo</value>
   </property>
   ```

For example:

```
<property>
     <name>instance.volumes</name>
     <value>hdfs://node-1.example.com:8020/apps/accumulo</value>
</property>
```

5. Add the configuration property `instance.zookeeper.host` to the `accumulo-site.xml` file. The value of this property should be a comma-separated list of ZooKeeper servers.

   In this “host” file each non-commented line is expected to be some host which should have a process running on it. The “masters” file contains hosts which should run the Accumulo Master process (only one host will be the active master, the rest will be hot-standbys) and the “slaves” file contains hosts which should run the Accumulo TabletServer process.

   For example:

   ```
   <property>
       <name>instance.zookeeper.host</name>
       <value>server1:2181,server2:2181,server3:2181</value>
   <property>
   ```

6. Change permissions to restrict access to the data directory to the Accumulo user:

   ```
   su - hdfs
   ```

   ```
   hadoop fs -chmod -R 700 /apps/accumulo
   ```

7. Change ownership of the data directory to the Accumulo user and group.

   ```
   su - hdfs
   ```

   ```
   hadoop fs -chown -R accumlo:accumulo /apps/accumulo
   ```

8. The example configuration files also include an accumulo-env.sh file.

   - If JAVA_HOME is not defined in the environment, you should specify it by editing the following line of code in the accumulo-env.sh file:

     ```
     test -z "$JAVA_HOME" && export JAVA_HOME=/path/to/java
     ```

     If you would like to prevent users from passing JAVA_HOME on the command line, remove the text prior to "export" and add the path to your JAVA_HOME. For example:

     ```
     export JAVA_HOME=/usr/hadoop-jdk1.7.0_67
     ```

   - If ZOOKEEPER_HOME is not defined in the environment, you should specify it by editing the following line of code in the accumulo-env.sh file:

     ```
     test -z "$ZOOKEEPER_HOME" && export ZOOKEEPER_HOME=/path/to/
     zookeeper
     ```

If you would like to prevent users from passing ZOOKEEPER_HOME on the command line, remove the text prior to "export" and add the path to your ZOOKEEPER_HOME. For example:

```
export ZOOKEEPER_HOME=/usr/hdp/current/zookeeper-client/conf
```

21.3. Configuring the "Hosts" Files

For multi-node systems, populate the following configuration files in $ACCUMULO_CONF_DIR: masters, slaves, tracers, monitor and gc. Each of these files corresponds to a list of hosts which run certain Accumulo processes. These files are for the Accumulo Master, TabletServer, Tracer, Monitor and GarbageCollector, respectively. In the provided example configurations, “localhost” is populated in these files. These files control the placement of Accumulo processes across many nodes.

When multiple hosts are specified in slaves and tracers, this will result in the appropriate process actively running on each of the listed hosts. For the other files, while the appropriate process will be started on each listed host, only one of the started processes will be active. The other processes will stay in a hot-standby state, attempting to become the active process that enables high-availability deployments.

21.4. Validating Accumulo

To validate that Accumulo is set up correctly:

1. Start the Accumulo service.

   a. Initialize Accumulo.

      ```
      /usr/hdp/current/accumulo-client/bin/accumulo init
      ```

   b. Enter a instance name and password.

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

      ```
      /usr/hdp/current/accumulo-client/bin/start-all.sh
      ```

2. View the Accumulo native UI.

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

   Look for any errors reported by the Accumulo monitor.

21.5. Smoke Testing Accumulo

Perform a smoke test to ensure that Accumulo is working properly by using the Accumulo shell.

1. Using the Accumulo shell, create a table in Accumulo:

   ```
   createtable testtable
   ```

2. Insert a row and assign a value:

   ```
   insert row colfam colqual value
   ```

3. Check to ensure the table exists:

   ```
   scan
   flush -w
   scan
   ```

4. Delete your test table:

   ```
   deletetable -f testtable
   ```

22. Installing Apache 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.

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.

22.1. Installing the Falcon Package

1. Install the Falcon RPM or package by using the following command:
   - RHEL/CentOS/Oracle Linux:

     ```
     yum install falcon
     ```
   - For SLES:

     ```
     zypper install falcon
     ```
2. After installation, verify that the owner and group of falcon.war in `/usr/hdp/current/falcon-server/webapp/` is falcon:falcon. If the owner and group are not falcon:falcon, change them using the following command:

   ```
   chown falcon:falcon falcon.war.
   ```
3. Update the falcon.url in `/etc/falcon/conf/client.properties` to use the web url of falcon server.
4. By default Falcon starts at port 15443. Set ".falcon.enableTLS" to false in `/etc/falcon/conf/startup-properties` to disable SSL and to start Falcon at port 15000.

22.2. Setting Directories and Permissions

1. Create directories and configure ownership and permissions as described below. Run the following commands:

   ```
   mkdir -p $FALCON_LOG_DIR;chown -R $FALCON_USER:$HADOOP_GROUP
    $FALCON_LOG_DIR;chmod -R 755 $FALCON_LOG_DIR;

   mkdir -p $FALCON_PID_DIR;chown -R $FALCON_USER:$HADOOP_GROUP
    $FALCON_PID_DIR;chmod -R 755 $FALCON_PID_DIR;

   mkdir -p $FALCON_DATA_DIR;chown -R $FALCON_USER:$HADOOP_GROUP
    $FALCON_DATA_DIR;chmod -R 755 $FALCON_DATA_DIR;

   mkdir -p <graph.storage.directory>;chown -R $FALCON_USER:$HADOOP_GROUP
    <graph.storage.directory>;chmod -R 755 <graph.storage.directory>;

   mkdir -p <config.store.uri>;chown -R $FALCON_USER:$HADOOP_GROUP <config.
   store.uri>;chmod -R 755 <config.store.uri>;
   ```

   where:

   $FALCON_LOG_DIR is the directory to store the Falcon logs. For example, `/var/log/falcon`.

   $FALCON_PID_DIR is the directory to store the Falcon process ID. For example, `/var/run/falcon`.

   $FALCON_DATA_DIR is the directory to store the falcon active mq data. For example, `/hadoop/falcon/embeddedmq/data`.

   <graph.storage.directory> is the graph storage directory defined in Falcon startup.properties key "*.falcon.graph.storage.directory". For example, `/usr/hdp/current/falcon-server/data/graphdb`.

   <config.store.uri> is the location to store user entity configurations defined in Falcon startup.properties key "*.config.store.uri". For example, `/usr/hdp/current/falcon-server/data/falcon-store/`.

   $FALCON_USER is the user owning the Falcon service. For example, falcon.

   $HADOOP_GROUP is a common groupshared by services. For example, hadoop.

2. For the Hive DR feature, the UI expects the template files to be copied to `/apps/data-mirroring` path. Create the following directories in HDFS:

   ```
   su - $FALCON_USER
   hdfs dfs -mkdir /apps/data-mirroring
   hdfs dfs -mkdir /apps/data-mirroring/workflows/
   hdfs dfs -copyFromLocal /usr/hdp/current/falcon-server/data-mirroring /apps
   hdfs dfs -chmod -R 770 /apps/data-mirroring
   hdfs dfs -chown -R falcon:users /apps/data-mirroring
   ```

   where:

   $FALCON_USER is the user owning the Falcon service. For example, falcon.

22.3. Configuring Proxy Settings

1. Stop all services. See Stopping HDP Services in the HDP Reference Guide for details.
2. Change the proxy settings for the falcon user in the core-site.xml file to allow falcon to impersonate users and groups:

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

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

   where:

   hadoop.proxyuser.falcon.groups is a comma-separated list of the UNIX groups whose users may be impersonated by Falcon

   hadoop.proxyuser.falcon.hosts is a comma-separated list of the hosts that are allowed to submit requests by Falcon
3. Start all Services. See Controlling HDP Services Manually in the HDP Reference Guide for details.

22.4. 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 shown in "Defining Data Pipelines" (in *Data Governance with Apache Falcon*), or create your own based on Falcon Schemas (also in the Data Governance guide).

22.5. 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. Run the following commands on the Oozie host machine.

   ```
   su - $OOZIE_USER
   ```

   ```
   /usr/hdp/current/oozie-server/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=$hadoopConfDir1,$NameNode2=$hadoopConfDir2,
$ResourceManager2:$rpcPortRM=$hadoopConfDir2,$NameNode3 :$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 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</value>
</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/hdp/current/oozie/oozie.war.` This will ensure that all existing items in the WAR file are still present after the current update.

   ```
   su - root
   ```

   ```
   cp $CATALINA_BASE/webapps/oozie.war /usr/hdp/current/oozie-
   server/oozie.war
   ```

   where $CATALINA_BASE is the path for the Oozie web app. By default, $CATALINA_BASE is:

   `/usr/hdp/2.3.0.0-<$version>/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/hdp/current/falcon-server/oozie/ext/falcon-oozie-el-
   extension-<$version>.jar /tmp/falcon-oozie-jars
   ```

   ```
   cp /tmp/falcon-oozie-jars/falcon-oozie-el-extension-<
   $version>.jar /usr/hdp/2.3.0.0-<$version>/oozie/libext
   ```

7. Package the Oozie WAR file as the Oozie user

```
su - $OOZIE_USER

cd /usr/hdp/current/oozie-server/bin

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

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

8. Start the Oozie service on all Falcon clusters. Run these commands on the Oozie host machine.

   ```
   su - $OOZIE_USER

   /usr/hdp/current/oozie-server/bin/oozie-start.sh
   ```

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

22.6. Configuring Hive for Falcon

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

Important

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 series of subdirectories `action-conf/hive`. Under the `hive` subdirectory, 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.

22.7. Configuring for Secure Clusters

If you are using secure clusters, complete the following steps.

1. Verify that `hadoop.security.auth_to_local in core-site.xml` is consistent across all clusters.

   ### Important

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

2. For working with secure clusters that use hive and hcatalog, the cluster.xml entity should have hadoop.rpc.protection set to the value of the hadoop cluster's hadoop.rpc.protection. For example:

   ```
   <property name="hadoop.rpc.protection" value="authentication"/>
   ```

   Note

   Value cannot be hard coded to authentication. It has to match the authentication value the hadoop cluster uses.

3. Set dfs.namenode.kerberos.principal for the cluster NameNode. For example:

   ```
   <property name="dfs.namenode.kerberos.principal" value="nn/
   ip-172-31-47-87.ec2.internal@EXAMPLE.COM"/>
   ```

4. For the hcatalog retention/replication/process to work with secure clusters, set hive.metastore.sasl.enabled to true in the cluster entity. For example:

   ```
   <property name="hive.metastore.sasl.enabled" value="true"/>
   ```

5. Set hive.metastore.kerberos.principal and hive.metastore.uris. For example:

   ```
   <property name="hive.metastore.kerberos.principal" value="hive/
   ip-172-31-47-87.ec2.internal@EXAMPLE.COM"/>
   <property name="hive.metastore.uris" value="thrift://ip-172-31-47-87.ec2.
   internal:9083"/>
   ```

6. For Windows Azure Storage Blob (WASB) replication to work, the target cluster's `core-site.xml` must have wasb credentials. For example:

   ```
   <property>
       <name>fs.azure.account.key.testuser.blob.core.windows.net</name>
       <value>XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX</value>
   </property>
   ```

7. Create the following property definitions in your cluster entity or entities. In the following example, replace $my.internal@EXAMPLE.COM and $my.internal with your own values.

   ```
   <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>
   ```

22.8. 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 submit $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 submit:

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

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

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

23. Installing Apache Knox

Apache Knox Gateway (Apache Knox) is the Web/REST API Gateway solution for Hadoop and provides a single access point for all of Hadoop resources over REST. The Knox Gateway 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 Package on the Knox server [187]
2. Set up and Validate the Knox Gateway Installation [187]

23.1. Install the Knox Package on the Knox server

To install the Knox RPM or package, 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/hdp/current/knox-server, referred to as $gateway_home`
- Knox configuration directory: `/etc/knox/conf`
- Knox log directory: `/var/log/knox`

23.2. 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 how to test access to your existing cluster with the minimal configuration.

Use the steps 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 will be prompted for the master secret. Enter the password at the prompt.

2. Start the gateway:

   ```
   su -l knox -c "/usr/hdp/current/knox-server/bin/gateway.sh
   start"
   ```

   ```
   Starting Gateway succeeded with PID 1871.
   ```

   The gateway starts. 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 "/usr/hdp/current/knox-server/bin/ldap.sh start"
   ```

   ```
   Starting LDAP succeeded with PID 1965.
   ```

   In a production environment, use Active Directory or OpenLDAP for authentication. For detailed instructions on configuring the 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 the gateway host to the WebHDFS Service host using telnet:

   Note

   To enable telnet set dfs.webhdfs.enabled to true.

   ```
   telnet $webhdfs_host $webhdfs_port
   ```

   Important

   You must be able to reach the internal cluster service from the machine on which Knox is running before continuing.

6. Update the WebHDFS host information and any other host and port in the topology to match your deployment.

   Note

   Your set up is not complete until all of the host:port information is updated.

The WebHDFS host information is located in the `$gateway_home/conf/topologies/sandbox.xml` file.

a. Find the 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.

b. (**Optional**) 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. (**Optional**) 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 -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.

24. Installing Apache Slider

Prerequisites

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

2. Verify the HDP repositories are available:

   ```
   yum list slider
   ```

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

   ```
   slider.noarch <version>
   ```

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

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

Installation

1. Run the following command to install Slider.

   - For RHEL/CentOS/Oracle Linux:

     ```
     yum install slider_2*
     ```

   - For SLES:

     ```
     zypper install slider_2*
     ```

2. As the root user, edit the following properties in the `/etc/hadoop/conf/yarn-site.xml` file.

   ```
   <property>
        <name>hadoop.registry.zk.quorum</name>
        <value>$ZOOKEEPERQUORUM-SERVERS</value>
        <description>List of hostname:port pairs defining the zookeeper quorum
    binding for the registry
        </description>
   </property>

   <property>
        <name>hadoop.registry.rm.enabled</name>
        <value>true</value>
        <description> Is the registry enabled: does the RM start it up, create
    the user
           and system paths, and purge service records when containers,
    application attempts
           and applications complete?
        </description>
   </property>
   ```

Set hadoop.registry.rm.enabled to true and set hadoop.registry.zk.quorum to the address and port number of your ZooKeeper Quorum server (usually assigned to port 2181). For example:

```
<property>
     <name>hadoop.registry.zk.quorum</name>
     <value>node-1.example.com:2181</value>
     <description>List of hostname:port pairs defining the zookeeper quorum
 binding for the registry
     </description>
</property>

<property>
     <name>hadoop.registry.rm.enabled</name>
     <value>true</value>
     <description>Is the registry enabled: does the RM start it up, create
 the user
        and system paths, and purge service records when containers,
 application attempts
        and applications complete?
     </description>
</property>
```

3. As the root user, specify the JAVA_HOME and HADOOP_CONF_DIR settings in the `/etc/slider/conf/slider-env.sh` file. For example:

```
# this is the shell script to start Slider deploying an application
# Usage: slider <action> <commands>

# The env variable SLIDER_JVM_OPTS can be used to override
# the default JVM opts

export JAVA_HOME=/usr/hadoop-jdk1.6.0_31
export HADOOP_CONF_DIR=/etc/hadoop/conf
```

4. Use the following command to switch to the slider bin directory:

```
cd /usr/hdp/current/slider-client/bin
```

5. Use the Slider version command to verify that Slider has installed properly:

```
./slider version
```

6. Ensure that there are no errors, and that your results say “Compiled against Hadoop <current_hadoop_version>”.

```
[root@node-1 bin]# ./slider version
2014-10-27 14:42:45,340 [main] INFO client.SliderClient - Slider Core-0.51.
0.2.3.0.0 Built against commit# d766e78d77 on Java 1.6.0_31 by jenkins
2014-10-27 14:42:45,351 [main] INFO client.SliderClient - Compiled against
 Hadoop 2.3.0.0-2800
2014-10-27 14:42:45,375 [main] INFO client.SliderClient - Hadoop runtime
 version (no branch) with source checksum 517963c273a1f4f8f5bfc15d92aa013
 and build date 2014-10-27T03:27Z
2014-10-27 14:42:45,383 [main] INFO util.ExitUtil - Exiting with status 0
[root@node-1 bin]#
```

25. Installing and Configuring Apache Atlas

This section describes how to install and configure Apache Atlas for HDP:

25.1. Atlas Prerequisites

Before installing Atlas, make sure your cluster meets the following prerequisites:

Table 25.1. Atlas Cluster Prerequisites

Item	Prerequisite
Cluster Stack Version	HDP 2.3.0 or later

25.2. Installing Atlas

To install Atlas, run the following commands as root:

- For RHEL or CentOS:

```
yum install atlas-metadata_2_3_*
```

- For SLES:

```
zypper install atlas-metadata_2_3_*
```

25.3. Installing Atlas Metadata Hive Plugin

To install Atlas Metadata Hive Plugin on the HiveServer2 Host, use the following command:

- For RHEL or CentOS:

```
yum install atlas-metadata_2_3_*-hive-plugin*
```

- For SLES:

```
zypper install atlas-metadata_2_3_*-hive-plugin*
```

Note

By default the config directory used by Atlas is `/usr/hdp/<hdp-version>/atlas/conf`. To override this set environment the variable METADATA_CONF to the path of the conf dir. atlas-env.sh is included in Atlas config directory. This file can be used to set various environment variables that you need for your services. In addition, you can set any other environment variables you might need.

25.4. Configuring Hive Hook

Atlas requires changes to `hive-site.xml` as well as additional configuration files as follows:

1. Create a new file in `/etc/hive/conf` named client.properties and another new file in `/etc/hive/conf/conf.server` named client.properties with the following contents:

   ```
   atlas.http.authentication.enabled=false|true
   ```

   ```
   atlas.http.authentication.type=simple|kerberos
   ```

2. Configure the Hive post execution hook by manually adding the following properties to hive-site.xml:

   ```
   <property>
         <name>atlas.cluster.name</name>
         <value>NAME-OF-CLUSTER</value>
   </property>

   <property>
         <name>atlas.rest.address</name>
         <value>http://ATLAS-FQDN:21000</value>
   </property>
   ```

3. Add the Atlas hook to list of any existing Hive post execution hooks. For example:

   ```
   <property>
         <name>hive.exec.post.hooks</name>
         <value>org.apache.hadoop.hive.ql.hooks.ATSHook, org.apache.atlas.hive.
   hook.HiveHook</value>
   </property>
   ```

25.5. Configuring for Secure Clusters

There are two sets of configuration properties required to secure the Atlas server: service identify properties and HTTP authentication properties.

1. Set the service identify properties.

   The following properties designate the authentication mechanism leveraged by the server to establish its identity:

   ```
   atlas.authentication.method = simple|kerberos
   ```

`atlas.authentication.principal = atlas/_HOST` (required if the method selected is "kerberos")

`atlas.authentication.keytab = <path to keytab file containing defined principal>`

2. Set the HTTP Authentication properties.

   The properties for configuring the Atlas server HTTP authentication mechanism are:

   `atlas.http.authentication.enabled = true|false`

   `atlas.http.authentication.type = simple|kerberos`

   `atlas.http.authentication.kerberos.principal = HTTP/_HOST` (required for Kerberos authentication type)

   `atlas.http.authentication.kerberos.keytab = <path to SPNEGO keytab>` (required for Kerberos authentication type)

3. Configure the Hive post execution hook by manually adding the following properties to `hive-site.xml`:

```
<property>
      <name>atlas.cluster.name</name>
      <value>NAME-OF-CLUSTER</value>
    </property>

    <property>
      <name>atlas.rest.address</name>
      <value>http://ATLAS-FQDN:21000</value>
    </property>
```

4. Add the Atlas hook to list of any existing Hive post execution hooks. For example:

```
<property>
      <name>hive.exec.post.hooks</name>
      <value>org.apache.hadoop.hive.ql.hooks.ATSHook, org.apache.atlas.hive.
hook.HiveHook</value>
    </property>
```

25.6. Validating Atlas

1. On Atlas Metadata Server host, run the following command to start ATLAS server:

   `/usr/hdp/<hdp-version>/atlas/bin/atlas_start.py –port 21000`

2. Open the Atlas Dashboard by visiting the following url:

   `http://<ATLAS_Metadata_Server_FQDN>:21000/`

3. To validate Atlas is running properly in an unsecure configuration, access the Atlas REST API by invoking the following command:

```
curl -sL -w "%{http_code} %{url_effective}\\n" "http://<ATLAS-HOST>:21000" -
o /dev/null
```

where ATLAS-HOST is the FQDN of the host where Atlas has been installed. The expected response should return a 200 response along with the URL that was accessed; similar to:

```
200 http://localhost:21000/
```

26. Setting Up Security for Manual Installs

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

26.1. Preparing Kerberos

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

26.1.1. 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 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 users and services (known as *principals*) 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 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.

26.1.2. 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, or 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
     ```

   Note

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

2. When the server is installed you must edit the two main configuration files.

   Update the KDC configuration by replacing EXAMPLE.COM with your domain and kerberos.example.com with the FQDN of the KDC host. Configuration files are in the following locations:

   - 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.

26.1.3. 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:

```
kdb5_util create -s
```

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. Start **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 assumes that 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
     ```

26.1.4. Creating Service Principals and Keytab Files for HDP

Each service in HDP must have its own principal. Because 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 create the principal, using mandatory naming conventions. Then create the keytab file with that principal's information, and copy the file to the keytab directory on the appropriate service host.

1. To create a service principal you will use the kadmin utility. This is a command-line driven utility into which you enter Kerberos commands to manipulate the central database. To start kadmin, enter:

   ```
   'kadmin $USER/admin@REALM'
   ```

To create a service principal, enter the following:

```
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.

The $principal_name part of the name must match the values in the following table.

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:

a. If the Kerberos credentials for one DataNode are compromised, it does not automatically lead to all DataNodes being compromised

b. 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.

Note: The NameNode, Secondary NameNode, and Oozie require two principals each.

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 NameNodes 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 26.1. 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*
Storm	Nimbus server DRPC daemon	nimbus/*$FQDN* **
Storm	Storm UI daemon Storm Logviewer daemon Nodes running process controller (such as Supervisor)	storm/*$FQDN* **
Kafka	KafkaServer	kafka/*$FQDN*
Hue	Hue Interface	hue/*$FQDN*
ZooKeeper	ZooKeeper	zookeeper/*$FQDN*
JournalNode Server*	JournalNode	jn/*$FQDN*
Gateway	Knox	knox/*$FQDN*

* Only required if you are setting up NameNode HA.

** For more information, see Configure Kerberos Authentication for Storm.

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

```
kadmin: addprinc -randkey dn/$datanode-host@$hadoop.realm
```

2. Extract the related keytab file and place it in the keytab directory of the appropriate respective components. The default directory is `/etc/krb5.keytab`.

```
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 the following table.

Table 26.2. 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

Component	Principal Name	Mandatory Keytab File Name
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
Storm	storm/*$FQDN*	storm.service.keytab
Kafka	kafka/*$FQDN*	kafka.service.keytab
Hue	hue/*$FQDN*	hue.service.keytab
ZooKeeper	zookeeper/*$FQDN*	zk.service.keytab
Journal Server*	jn/*$FQDN*	jn.service.keytab
Knox Gateway**	knox/*$FQDN*	knox.service.keytab

* Only required if you are setting up NameNode HA.

** 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.

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.

26.2. Configuring HDP

This section provides information on configuring HDP for Kerberos.

- Configuration Overview [201]
- Creating Mappings Between Principals and UNIX Usernames [202]
- Examples [203]
- Adding Security Information to Configuration Files [203]

26.2.1. Configuration Overview

Note

You must adhere to the existing upper and lower case naming conventions in the configuration file templates.

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.

26.2.2. 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.

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.

- **The Base**

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

  For example:

  ```
  [1:$1@$0] translates myusername@APACHE.ORG to myusername@APACHE.ORG
  [2:$1] translates myusername/admin@APACHE.ORG to myusername
  ```

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

- **The Filter**

  The filter consists of a regular expression (regex) in a parentheses. It must match the generated string for the rule to apply.

  For example:

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

- **The Substitution**

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

  ```
  s/@ACME\.COM// removes the first instance of @ACME.DOMAIN
  s/@[A-Z]*\.COM// remove the first instance of @ followed by a name followed
   by COM.
  s/X/Y/g replace all of X's in the name with Y
  ```

26.2.3. 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, the following rule would do this:

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

- To translate names with a second component, you could 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
  ```

26.2.4. 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
  ```

26.2.4.1. core-site.xml

Add the following information to the `core-site.xml` file on *every* host in your cluster:

Table 26.3. General core-site.xml, Knox, and Hue

Property Name	Property Value	Description
hadoop.security.authentication	kerberos	Set the authentication type for the cluster. Valid values are: simple or kerberos.
hadoop.rpc.protection	authentication; integrity; privacy	This is an [OPTIONAL] setting. If not set, defaults to authentication. authentication = authentication only; the client and server mutually authenticate during connection setup. integrity = authentication and integrity; guarantees the integrity of data exchanged between client and server as well as authentication. privacy = authentication, integrity, and confidentiality; guarantees that data exchanged between client and server is encrypted and is not readable by a “man in the middle”.
hadoop.security.authorization	true	Enable authorization for different protocols.
hadoop.security.auth_to_local	The mapping rules. For example: `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.

Following is 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 26.4. core-site.xml Master Node Settings – Knox Gateway

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.

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

Table 26.5. core-site.xml Master Node Settings – Hue

Property Name	Property Value	Description
hue.kerberos.principal.shortname	hue	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example cli*.
hadoop.proxyuser.hue.groups	*	Group to which all the Hue users belong. Use the wild card character to select multiple groups, for example cli*.
hadoop.proxyuser.hue.hosts	*	
hadoop.proxyuser.knox.hosts	$hue_host_FQDN	Identifies the Knox Gateway host.

Following is the XML for both Knox and Hue settings:

```
<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>
```

26.2.4.2. hdfs-site.xml

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

Table 26.6. hdfs-site.xml File Property Settings

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 behavioris unchanged. Switching from one parameter value to the other does not change the mode, owner or group of files or directories.
dfs.permissions.supergroup	hdfs	The name of the group of super-users.
dfs.block.access.token.enable	true	If true, access tokens are used as capabilities for accessing DataNodes. If false, no access tokens are checked on accessing DataNodes.
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the NameNode.
dfs.secondary.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM	Kerberos principal name for the secondary NameNode.
dfs.web.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM	The HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint. The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP SPNEGO specification.
dfs.web.authentication.kerberos.keytab	/etc/security/keytabs/spnego.service.keytab	The Kerberos keytab file with the credentials for the HTTP Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM	The Kerberos principal that the DataNode runs as. "_HOST" is replaced by the real host name.
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals.
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab	Combined keytab file containing the NameNode service and host principals. <question?>
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab	The filename of the keytab file for the DataNode.
dfs.https.port	50470	The HTTPS port to which the NameNode binds.
dfs.namenode.https-address	Example: ip-10-111-59-170.ec2.internal:50470	The HTTPS address to which the NameNode binds.
dfs.datanode.data.dir.perm	750	The permissions that must be set on the dfs.data.dir directories. The DataNode will not come up if all existing dfs.data.dir directories do not

Property Name	Property Value	Description
		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}	

Following is 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
```

26.2.4.3. yarn-site.xml

You must add the following information to the `yarn-site.xml` file on *every* host in your cluster:

Table 26.7. yarn-site.xml Property Settings

Property	**Value**	**Description**
yarn.resourcemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the ResourceManager.
yarn.resourcemanager.keytab	/etc/krb5.keytab	The keytab for the ResourceManager.
yarn.nodemanager.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the NodeManager.
yarn.nodemanager.keytab	/etc/krb5.keytab	The keytab for the NodeManager.
yarn.nodemanager.container-executor.class	org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor	The class that will execute (launch) the containers.
yarn.nodemanager.linux-container-executor.path	hadoop-3.0.0-SNAPSHOT/bin/container-executor	The path to the Linux container executor.
yarn.nodemanager.linux-container-executor.group	hadoop	A special group (e.g., hadoop) with executable permissions for the container executor, of which the NodeManager UNIX user is the group member and no ordinary application user is. If any application user belongs to this special group, security will be compromised. This special group name should be specified for the configuration property.
yarn.timeline-service.principal	yarn/localhost@EXAMPLE.COM	The Kerberos principal for the Timeline Server.
yarn.timeline-service.keytab	/etc/krb5.keytab	The Kerberos keytab for the Timeline Server.
yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled	true	Flag to enable override of the default Kerberos authentication filter with the RM authentication filter to allow authentication using delegation tokens (fallback to Kerberos if the tokens are missing). Only applicable when the http authentication type is Kerberos.
yarn.timeline-service.http-authentication.type	kerberos	Defines authentication used for the Timeline Server HTTP endpoint. Supported values are: simple | kerberos |

Property	Value	Description
		$AUTHENTICATION_HANDLER _CLASSNAME
yarn.timeline-service.http-authentication.kerberos.principal	HTTP/localhost@EXAMPLE.COM	The Kerberos principal to be used for the Timeline Server HTTP endpoint.
yarn.timeline-service.http-authentication.kerberos.keytab	authentication.kerberos.keytab /etc/krb5.keytab	The Kerberos keytab to be used for the Timeline Server HTTP endpoint.

Following is the XML for these entries:

```
<property>
     <name>yarn.resourcemanager.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.resourcemanager.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.nodemanager.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.nodemanager.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.nodemanager.container-executor.class</name>
     <value>org.apache.hadoop.yarn.server.nodemanager.LinuxContainerExecutor</
value>
</property>

<property>
     <name>yarn.nodemanager.linux-container-executor.path</name>
     <value>hadoop-3.0.0-SNAPSHOT/bin/container-executor</value>
</property>

<property>
     <name>yarn.nodemanager.linux-container-executor.group</name>
     <value>hadoop</value>
</property>

<property>
     <name>yarn.timeline-service.principal</name>
     <value>yarn/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.timeline-service.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>

<property>
     <name>yarn.resourcemanager.webapp.delegation-token-auth-filter.enabled</
name>
     <value>true</value>
```

```
</property>

<property>
     <name>yarn.timeline-service.http-authentication.type</name>
     <value>kerberos</value>
</property>

<property>
     <name>yarn.timeline-service.http-authentication.kerberos.principal</name>
     <value>HTTP/localhost@EXAMPLE.COM</value>
</property>

<property>
     <name>yarn.timeline-service.http-authentication.kerberos.keytab</name>
     <value>/etc/krb5.keytab</value>
</property>
```

26.2.4.4. mapred-site.xml

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

Table 26.8. mapred-site.xml Property Settings

Property Name	Property Value	Description
mapreduce.jobhistory.keytab	/etc/security/keytabs/jhs.service.keytab	Kerberos keytab file for the MapReduce JobHistory Server.
mapreduce.jobhistory.principal	jhs/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the MapReduce JobHistory Server.
mapreduce.jobhistory.webapp.address	TODO-JOBHISTORYNODE-HOSTNAME:19888	MapReduce JobHistory Server Web UI host:port
mapreduce.jobhistory.webapp.https.address	TODO-JOBHISTORYNODE-HOSTNAME:19889	MapReduce JobHistory Server HTTPS Web UI host:port
mapreduce.jobhistory.webapp.spnego-keytab-file	/etc/security/keytabs/spnego.service.keytab	Kerberos keytab file for the spnego service.
mapreduce.jobhistory.webapp.spnego-principal	HTTP/_HOST@TODO-KERBEROS-DOMAIN	Kerberos principal name for the spnego service.

Following is the XML for these entries:

```
<property>
     <name>mapreduce.jobhistory.keytab</name>
     <value>/etc/security/keytabs/jhs.service.keytab</value>
</property>

<property>
     <name>mapreduce.jobhistory.principal</name>
     <value>jhs/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.address</name>
     <value>TODO-JOBHISTORYNODE-HOSTNAME:19888</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.https.address</name>
     <value>TODO-JOBHISTORYNODE-HOSTNAME:19889</value>
```

```
</property>

<property>
     <name>mapreduce.jobhistory.webapp.spnego-keytab-file</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
</property>

<property>
     <name>mapreduce.jobhistory.webapp.spnego-principal</name>
     <value>HTTP/_HOST@TODO-KERBEROS-DOMAIN</value>
</property>
```

26.2.4.5. hbase-site.xml

For HBase to run on a secured cluster, HBase must be able to authenticate itself to HDFS. Add the following information to the `hbase-site.xml` file on your HBase server. There are no default values; the following are only examples:

Table 26.9. hbase-site.xml Property Settings – HBase Server

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. 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

Property Name	Property Value	Description
		MasterObserver, just put it in HBase's classpath and add the fully qualified class name here.

Following is 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>
```

26.2.4.6. hive-site.xml

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

To the `hive-site.xml` file on every host in your cluster, add the following information:

Table 26.10. hive-site.xml Property Settings

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.

Following is 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>
```

26.2.4.7. oozie-site.xml

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

Table 26.11. oozie-site.xml Property Settings

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.principaloozie/_HOSTl@EXAMPLE.COM	oozie/_HOSTl@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		

Property Name	Property Value	Description
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.

26.2.4.8. webhcat-site.xml

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

Table 26.12. webhcat-site.xml Property Settings

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	
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.

26.2.4.9. limits.conf

Adjust the Maximum Number of Open Files and Processes

In a secure cluster, if the DataNodes are started as the root user, JSVC downgrades the processing using setuid to hdfs. However, the ulimit is based on the ulimit of the root user, and the default ulimit values assigned to the root user for the maximum number of open files and processes may be too low for a secure cluster. This can result in a “Too Many Open Files” exception when the DataNodes are started.

Therefore, when configuring a secure cluster you should increase the following root ulimit values:

- nofile: The maximum number of open files. Recommended value: 32768
- nproc: The maximum number of processes. Recommended value: 65536

To set system-wide ulimits to these values, log in as root and add the following lines to the the `/etc/security/limits.conf` file on *every* host in your cluster:

```
* - nofile 32768
* - nproc 65536
```

To set only the root user ulimits to these values, log in as root and add the following lines to the the `/etc/security/limits.conf` file.

```
root - nofile 32768
root - nproc 65536
```

You can use the ulimit -a command to view the current settings:

```
[root@node-1 /]# ulimit -a
core file size (blocks, -c) 0
data seg size (kbytes, -d) unlimited
scheduling priority (-e) 0
file size (blocks, -f) unlimited
pending signals (-i) 14874
max locked memory (kbytes, -l) 64
max memory size (kbytes, -m) unlimited
open files (-n) 1024
pipe size (512 bytes, -p) 8
POSIX message queues (bytes, -q) 819200
real-time priority (-r) 0
stack size (kbytes, -s) 10240
cpu time (seconds, -t) unlimited
max user processes (-u) 14874
virtual memory (kbytes, -v) unlimited
file locks (-x) unlimited
```

You can also use the ulimit command to dynamically set these limits until the next reboot. This method sets a temporary value that will revert to the settings in the `/etc/security/limits.conf` file after the next reboot, but it is useful for experimenting with limit settings. For example:

```
[root@node-1 /]# ulimit -n 32768
```

The updated value can then be displayed:

```
[root@node-1 /]# ulimit -n
32768
```

26.3. Configuring Hue

To enable Hue to work with a HDP cluster configured for Kerberos, make the following changes to Hue and Kerberos.

1. Configure Kerberos as described in Setting Up Security for Manual Installs.
2. Create a principal for the Hue Server.

   ```
   addprinc -randkey hue/$FQDN@EXAMPLE.COM
   ```

   where $FQDN is the hostname of the Hue Server and EXAMPLE.COM is the Hadoop realm.
3. Generate a keytab for the Hue principal.

   ```
   xst -k hue.service.keytab hue/$FQDN@EXAMPLE.COM
   ```
4. Place the keytab file on the Hue Server. Set the permissions and ownership of the keytab file.

```
/etc/security/keytabs/hue.service.keytab
chown hue:hadoop /etc/security/keytabs/hue.service.keytab
chmod 600 /etc/security/keytabs/hue.service.keytab
```

5. Confirm the keytab is accessible by testing kinit.

```
su - hue kinit -k -t /etc/security/keytabs/hue.service.keytab hue/
$FQDN@EXAMPLE.COM
```

6. Add the following to the [kerberos] section in the `/etc/hue/conf/hue.ini` configuration file.

```
[[kerberos]]
# Path to Hue's Kerberos keytab file
hue_keytab=/etc/security/keytabs/hue.service.keytab
# Kerberos principal name for Hue
hue_principal=hue/$FQDN@EXAMPLE.COM
```

7. Set the path to the kinit based on the OS.

```
# Path to kinit
# For RHEL/CentOS 6.x, kinit_path is /usr/bin/kinit
kinit_path=/usr/kerberos/bin/kinit
```

8. Set security_enabled=true for every component in hue.ini.

```
[[hdfs_clusters]], [[yarn_clusters]], [[liboozie]], [[hcatalog]]
```

9. Save the hue.ini file.

10. Restart Hue:

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

26.4. 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.

Important

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

26.4.1. 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. Mismatched encryption types cause 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.

26.4.2. 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:

   - To libdefaults, add the following properties.

     - Set 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
  ```

- Add a realm for the AD domain:

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

- Save the krb5.conf 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
   ```

   When defining encryption, be sure to also enter the encryption type (e.g., 'normal')

27. Uninstalling HDP

Use the following instructions to uninstall HDP:

1. Stop all of the installed HDP services. See Stopping HDP Services in the HDP Reference Guide.
2. If Knox is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove knox*
     ```
   - For SLES:

     ```
     zypper remove knox\*
     ```
3. If Ranger is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove ranger\*
     ```
   - For SLES:

     ```
     zypper remove ranger\*
     ```
4. If Kafka is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove kafka\*
     ```
   - For SLES:

     ```
     zypper remove kafka\*
     ```
5. If Storm is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove storm\*
     ```
   - For SLES:

     ```
     zypper remove storm\*
     ```
6. If Hive is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove hive\*
     ```
   - For SLES:

```
zypper remove hive\*
```

7. If HBase is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove hbase\*
     ```
   - For SLES:

     ```
     zypper remove hbase\*
     ```
8. If Phoenix is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove phoenix\*
     ```
   - For SLES:

     ```
     zypper remove phoenix\*
     ```
9. If Accumulo is installed, run the following command on all the cluster nodes:
   - For RHEL/CentOS/Oracle Linux:

     ```
     yum remove accumulo\*
     ```
   - For SLES:

     ```
     zypper remove accumulo\*
     ```
10. If Tez is installed, run the following command on all the cluster nodes:
    - For RHEL/CentOS/Oracle Linux:

      ```
      yum remove tez\*
      ```
    - For SLES:

      ```
      zypper remove tez\*
      ```
11. If ZooKeeper is installed, run the following command on all the cluster nodes:
    - For RHEL/CentOS/Oracle Linux:

      ```
      yum remove zookeeper\*
      ```
    - For SLES:

      ```
      zypper remove zookeeper\*
      ```
12. If Oozie is installed, run the following command on all the cluster nodes:
    - For RHEL/CentOS/Oracle Linux:

```
yum remove oozie\*
```

- For SLES:

```
zypper remove oozie\*
```

13. If Pig is installed, run the following command on all the cluster nodes:

- For RHEL/CentOS/Oracle Linux:

```
yum remove pig\*
```

- For SLES:

```
zypper remove pig\*
```

14. If compression libraries are installed, run the following command on all the cluster nodes:

```
yum remove snappy\* yum remove hadooplzo\*
```

15. If Knox is installed, run the following command on all the gateway host:

- For RHEL/CentOS/Oracle Linux:

```
yum remove knox\*
```

- For SLES:

```
zypper remove knox\*
```

16. Uninstall Hadoop. run the following command on all the cluster nodes:

```
yum remove hadoop\*
```

17. Uninstall ExtJS libraries and MySQL connector. Run the following command on all the cluster nodes:

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