Hortonworks Data Platform

Command Line Installation

(Aug 29, 2016)

docs.hortonworks.com

Hortonworks Data Platform: Command Line Installation

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

1. Preparing to Manually Install HDP 1
  1.1. Meeting Minimum System Requirements 1
    1.1.1. Hardware Recommendations 1
    1.1.2. Operating System Requirements 1
    1.1.3. Software Requirements 1
    1.1.4. JDK Requirements 2
    1.1.5. Metastore Database Requirements 5
  1.2. Virtualization and Cloud Platforms 14
  1.3. Configuring Remote Repositories 15
  1.4. Deciding on a Deployment Type 15
  1.5. Collect Information 15
  1.6. Prepare the Environment 16
    1.6.1. Enable NTP on Your Cluster 16
    1.6.2. Disable SELinux 17
    1.6.3. Disable IPTables 18
  1.7. Download Companion Files 19
  1.8. Define Environment Parameters 19
  1.9. Creating System Users and Groups 25
  1.10. Determining HDP Memory Configuration Settings 26
    1.10.1. Running the YARN Utility Script 26
    1.10.2. Calculating YARN and MapReduce Memory Requirements 27
  1.11. Configuring NameNode Heap Size 30
  1.12. Allocating Adequate Log Space for HDP 31
  1.13. Downloading the HDP Maven Artifacts 31
2. Installing Apache ZooKeeper 33
  2.1. Install the ZooKeeper Package 33
  2.2. Securing Zookeeper with Kerberos (optional) 34
  2.3. Securing ZooKeeper Access 34
    2.3.1. ZooKeeper Configuration 35
    2.3.2. YARN Configuration 36
    2.3.3. HDFS Configuration 36
  2.4. Set Directories and Permissions 37
  2.5. Set Up the Configuration Files 38
  2.6. Start ZooKeeper 39
3. Installing HDFS, YARN, and MapReduce 40
  3.1. Set Default File and Directory Permissions 40
  3.2. Install the Hadoop Packages 40
  3.3. Install Compression Libraries 41
    3.3.1. Install Snappy 41
    3.3.2. Install LZO 41
  3.4. Create Directories 41
    3.4.1. Create the NameNode Directories 42
    3.4.2. Create the SecondaryNameNode Directories 42
    3.4.3. Create DataNode and YARN NodeManager Local Directories 42
    3.4.4. Create the Log and PID Directories 43
    3.4.5. Symlink Directories with hdp-select 45
4. Setting Up the Hadoop Configuration 47
5. Validating the Core Hadoop Installation 53

5.1. Format and Start HDFS 53
5.2. Smoke Test HDFS 53
5.3. Configure YARN and MapReduce 54
5.4. Start YARN 56
5.5. Start MapReduce JobHistory Server 56
5.6. Smoke Test MapReduce 57
6. Installing Apache HBase 58
6.1. Install the HBase Package 58
6.2. Set Directories and Permissions 59
6.3. Set Up the Configuration Files 59
6.4. Validate the Installation 61
6.5. Starting the HBase Thrift and REST Servers 62
7. Installing Apache Phoenix 64
7.1. Installing the Phoenix Package 64
7.2. Configuring HBase for Phoenix 64
7.3. Configuring Phoenix to Run in a Secure Cluster 65
7.4. Validating the Phoenix Installation 66
7.5. Troubleshooting Phoenix 68
8. Installing and Configuring Apache Tez 69
8.1. Prerequisites 69
8.2. Installing the Tez Package 69
8.3. Configuring Tez 70
8.4. Creating a New Tez View Instance 76
8.5. Validating the Tez Installation 76
8.6. Troubleshooting 77
9. Installing Apache Hive and Apache HCatalog 78
9.1. Installing the Hive-HCatalog Package 78
9.2. Setting Directories and Permissions 80
9.3. Setting Up the Hive/HCatalog Configuration Files 80
9.3.1. HDP-Utility script 82
9.3.2. Configure Hive and HiveServer2 for Tez 82
9.4. Setting Up the Database for the Hive Metastore 84
9.5. Setting up RDBMS for use with Hive Metastore 86
9.6. Creating Directories on HDFS 87
9.7. Enabling Tez for Hive Queries 88
9.8. Disabling Tez for Hive Queries 88
9.9. Configuring Tez with the Capacity Scheduler 88
9.10. Validating Hive-on-Tez Installation 89
9.11. Installing Hive LLAP 90
9.12. Prerequisites 90
9.13. Preparing to Install LLAP 91
9.14. Installing LLAP on an Unsecured Cluster 91
9.15. Installing LLAP on a Secured Cluster 93
9.15.1. Prerequisites 93
9.15.2. Installing LLAP on a Secured Cluster 93
9.15.3. Validating the Installation on a Secured Cluster 99
9.16. Stopping the LLAP Service 100
9.17. Tuning LLAP for Performance 100
10. Installing Apache Pig 101
10.1. Install the Pig Package 101
10.2. Validate the Installation 101

11. Installing Apache WebHCat ... 102
- 11.1. Install the WebHCat Package ... 102
- 11.2. Upload the Pig, Hive and Sqoop tarballs to HDFS ... 102
- 11.3. Set Directories and Permissions ... 103
- 11.4. Modify WebHCat Configuration Files ... 104
- 11.5. Set Up HDFS User and Prepare WebHCat Directories ... 105
- 11.6. Validate the Installation ... 106

12. Installing Apache Oozie ... 107
- 12.1. Install the Oozie Package ... 107
- 12.2. Set Directories and Permissions ... 110
- 12.3. Set Up the Oozie Configuration Files ... 111
  - 12.3.1. For Derby ... 112
  - 12.3.2. For MySQL ... 113
  - 12.3.3. For PostgreSQL ... 114
  - 12.3.4. For Oracle ... 115
- 12.4. Configure Your Database for Oozie ... 117
- 12.5. Set up the Sharelib ... 118
- 12.6. Validate the Installation ... 118
- 12.7. Stop and Start Oozie ... 119

13. Installing Apache Ranger ... 120
- 13.1. Installation Prerequisites ... 120
- 13.2. Installing Policy Manager ... 120
  - 13.2.1. Install the Ranger Policy Manager ... 121
  - 13.2.2. Install the Ranger Policy Administration Service ... 123
  - 13.2.3. Start the Ranger Policy Administration Service ... 123
  - 13.2.4. Configuring the Ranger Policy Administration Authentication Mode ... 124
  - 13.2.5. Configuring Ranger Policy Administration High Availability ... 125
- 13.3. Installing UserSync ... 125
  - 13.3.1. Using the LDAP Connection Check Tool ... 125
  - 13.3.2. Install UserSync and Start the Service ... 131
- 13.4. Installing Ranger Plug-ins ... 134
  - 13.4.1. Installing the Ranger HDFS Plug-in ... 135
  - 13.4.2. Installing the Ranger YARN Plug-in ... 138
  - 13.4.3. Installing the Ranger Kafka Plug-in ... 141
  - 13.4.4. Installing the Ranger HBase Plug-in ... 144
  - 13.4.5. Installing the Ranger Hive Plug-in ... 147
  - 13.4.6. Installing the Ranger Knox Plug-in ... 150
  - 13.4.7. Installing the Ranger Storm Plug-in ... 155
- 13.5. Installing and Configuring Tag-Based Access Control ... 158
  - 13.5.1. Installing the Tag Synchronization Module ... 158
  - 13.5.2. Configuring Kafka Notifications ... 160
  - 13.5.3. Validating the Tag-Sync Module Installation ... 160
  - 13.5.4. Optional: Updating Password for Ranger Admin User rangertagsync ... 160
- 13.6. Verifying the Installation ... 161

14. Installing Hue ... 162
- 14.1. Before You Begin ... 162
- 14.2. Configure HDP to Support Hue ... 163
- 14.3. Install the Hue Packages ... 173
- 14.4. Configure Hue to Communicate with the Hadoop Components ... 174

14.4.1. Configure the Web Server ... 174
14.4.2. Configure Hadoop ... 175
14.5. Configure Hue for Databases ... 177
14.5.1. Using Hue with Oracle ... 177
14.5.2. Using Hue with MySQL ... 178
14.5.3. Using Hue with PostgreSQL ... 179
14.6. Start, Stop, and Restart Hue ... 180
14.7. Validate the Hue Installation ... 180
15. Installing Apache Sqoop ... 181
15.1. Install the Sqoop Package ... 181
15.2. Set Up the Sqoop Configuration ... 182
15.3. Validate the Sqoop Installation ... 182
16. Installing Apache Mahout ... 183
16.1. Install Mahout ... 183
16.2. Validate Mahout ... 183
17. Installing and Configuring Apache Flume ... 185
17.1. Understanding Flume ... 185
17.2. Installing Flume ... 186
17.3. Configuring Flume ... 188
17.4. Starting Flume ... 189
17.5. HDP and Flume ... 190
17.6. A Simple Example ... 190
18. Installing and Configuring Apache Storm ... 192
18.1. Install the Storm Package ... 192
18.2. Configure Storm ... 193
18.3. Configure a Process Controller ... 194
18.4. (Optional) Configure Kerberos Authentication for Storm ... 195
18.5. (Optional) Configuring Authorization for Storm ... 198
18.6. Validate the Installation ... 201
19. Installing and Configuring Apache Spark ... 203
19.1. Spark Prerequisites ... 203
19.2. Installing Spark ... 203
19.3. Configuring Spark ... 204
19.4. (Optional) Starting the Spark Thrift Server ... 207
19.5. (Optional) Configuring Dynamic Resource Allocation ... 208
19.6. Validating Spark ... 208
20. Installing and Configuring Apache Kafka ... 209
20.1. Install Kafka ... 209
20.2. Configure Kafka ... 210
20.3. Validate Kafka ... 211
21. Installing and Configuring Zeppelin ... 212
21.1. Installation Prerequisites ... 212
21.2. Installing the Zeppelin Package ... 212
21.3. Configuring Zeppelin ... 213
21.4. Starting, Stopping, and Restarting Zeppelin ... 214
21.5. Validating Zeppelin ... 214
21.6. Accessing the Zeppelin UI ... 214
22. Installing Apache Accumulo ... 216
22.1. Installing the Accumulo Package ... 216
22.2. Configuring Accumulo ... 217
22.3. Configuring the "Hosts" Files ... 219

22.4. Validating Accumulo 219
22.5. Smoke Testing Accumulo 219
23. Installing Apache Falcon 221
23.1. Installing the Falcon Package 221
23.2. Setting Directories and Permissions 222
23.3. Configuring Proxy Settings 223
23.4. Configuring Falcon Entities 223
23.5. Configuring Oozie for Falcon 223
23.6. Configuring Hive for Falcon 228
23.7. Configuring for Secure Clusters 228
23.8. Validate Falcon 230
24. Installing Apache Knox 231
24.1. Install the Knox Package on the Knox Server 231
24.2. Set up and Validate the Knox Gateway Installation 231
24.3. Configuring Knox Single Sign-on (SSO) 233
25. Installing Apache Slider 237
26. Installing and Configuring Apache Atlas 240
26.1. Atlas Prerequisites 240
26.2. Installing Atlas 240
26.3. Installing Atlas Metadata Apache Hive Plugin 241
26.4. Configuring Apache Hive Hook 241
26.5. Configuring Atlas For Apache Kafka Notifications 243
26.6. Configuring Apache Storm Hook 243
26.7. Configuring the Graph Database 244
26.7.1. Choosing Between Storage Backends 244
26.7.2. Choosing Between Indexing Backends 244
26.7.3. Configure Atlas to Use Apache HBase 245
26.7.4. Configure Atlas to Use SolrCloud 248
26.8. Configuring Apache Falcon Hook 249
26.9. Configuring Apache Sqoop Hook 250
26.10. Configuring for Secure Clusters 251
26.11. Configuring Atlas in a Kerberized Cluster 252
26.12. Validating Atlas 253
27. Setting Up Kerberos Security for Manual Installs 254
28. Uninstalling HDP 255

List of Tables

1.1. Directories Needed to Install Core Hadoop ... 19
1.2. Directories Needed to Install Ecosystem Components ... 20
1.3. Define Users and Groups for Systems ... 24
1.4. Typical System Users and Groups ... 25
1.5. yarn-utils-py Options ... 26
1.6. Reserved Memory Recommendations ... 28
1.7. Recommended Container Size Values ... 28
1.8. YARN and MapReduce Configuration Values ... 29
1.9. Example Value Calculations Without HBase ... 29
1.10. Example Value Calculations with HBase ... 30
1.11. Recommended NameNode Heap Size Settings ... 30
8.1. Tez Configuration Parameters ... 70
9.1. Hive Configuration Parameters ... 83
9.2. ... 90
9.3. ... 91
9.4. LLAP Properties to Set in hive-site.xml ... 92
9.5. HiveServer2 Properties to Set in hive-site.xml to Enable Concurrent Queries with LLAP ... 92
9.6. Properties to Set in hive-site.xml for Secured Clusters ... 96
9.7. Properties to Set in ssl-server.xml for LLAP on Secured Clusters ... 97
9.8. LLAP Package Parameters ... 98
11.1. Hadoop core-site.xml File Properties ... 103
13.1. install.properties Entries ... 121
13.2. Properties to Update in the install.properties File ... 132
13.3. Properties to Edit in the install.properties File ... 136
13.4. Properties to Edit in the install.properties File ... 138
13.5. Properties to Edit in the install.properties File ... 141
13.6. HBase Properties to Edit in the `install.properties` file ... 144
13.7. Hive-Related Properties to Edit in the install.properties File ... 147
13.8. Knox-Related Properties to Edit in the install.properties File ... 151
13.9. Storm-Related Properties to Edit in the `install.properties` file ... 155
13.10. install.properties File Properties ... 159
13.11. Run-time Directory Structure ... 159
17.1. Flume 1.5.2 Dependencies ... 187
18.1. Required jaas.conf Sections for Cluster Nodes ... 196
18.2. Supported Authorizers ... 199
18.3. storm.yaml Configuration File Properties ... 199
18.4. worker-launcher.cfg File Configuration Properties ... 200
18.5. multitenant-scheduler.yaml Configuration File Properties ... 200
19.1. Prerequisites for running Spark 1.6 ... 203
20.1. Kafka Configuration Properties ... 210
21.1. Installation Prerequisites ... 212
26.1. Atlas Cluster Prerequisites ... 240

1. Preparing to Manually Install HDP

This chapter describes how to prepare to install Hortonworks Data Platform (HDP) manually. You must complete the following tasks before you deploy Hadoop cluster using HDP:

1. Meet minimum requirements
2. Configure the remote repositories
3. Decide on deployment type
4. Collect information
5. Prepare the environment
6. Download companion files
7. Define environment parameters
8. [Optional] Create System Users and Groups
9. Determine HDP memory configuration settings
10. Allocate adequate log space for HDP

1.1. Meeting Minimum System Requirements

To run 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 2.5.0 consumes about 6.5 GB of disk space. For more information about HDP hardware recommendations, see the *HDP Cluster Planning Guide*.

1.1.2. Operating System Requirements

Refer to the HDP 2.5.0 Release Notes for information on supported operating systems.

1.1.3. Software Requirements

You must install the following software on each of your hosts:

- `apt-get` (for Ubuntu and Debian)
- `chkconfig` (Ubuntu and Debian)
- `curl`
- `php_curl` (for SLES)
- `reposync` (might not be installed by default on all SLES hosts)

- `rpm` (for RHEL, CentOS, or SLES)
- `scp`
- `tar`
- `unzip`
- `wget`
- `yum` (for RHEL or CentOS)
- `zypper` (for SLES)

In addition, if you are creating local mirror repositories as part of the installation process and you are using RHEL, CentOS, or SLES), then on the mirror repo server you need the following utilities:

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

See Deploying HDP in Production Data Centers with Firewalls.

1.1.4. JDK Requirements

Your system must have the correct Java Development Kit (JDK) installed on all cluster nodes. Refer to the HDP 2.5.0 Release Notes for information on supported JDKs.

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. Manually Installing Oracle JDK

Use the following instructions to manually install Oracle JDK:

1. If you do not have a `/usr/java` directory, 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.

3. 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.

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

5. 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 are extracted into a `/usr/java/jdk1.7.0_67` directory or a `/usr/java/jdk1.8.0_51` directory.

6. 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
   ```

7. Set the JAVA_HOME and PATH environment variables:

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

8. Verify that Java is installed in your environment:

   ```
   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. Manually Installing OpenJDK 7

OpenJDK7 on HDP 2.5.0 does not work if you are using SLES as your OS. Use the following instructions to manually install OpenJDK 7 on a Linux OS other than SLES:

1. Check your currently installed version from a terminal window:

   ```
   java -version
   ```

2. If the JDK version is earlier than 7, uninstall it, as in the following example using CentOS:

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

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

3. If you followed Step 2, verify that Java is uninstalled:

   ```
   which java
   ```

4. (Optional) Download OpenJDK 7 RPMs from the appropriate command line:

   RedHat, CentOS, or Oracle Linux:

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

Ubuntu or Debian:

```
apt-get install openjdk-7-jdk
```

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 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 that Java is installed in your environment by running the following command 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. Manually Installing the JCE

Unless you are using OpenJDK with unlimited-strength JCE, you must manually install the Java Cryptography Extension (JCE) security policy files on all hosts in the cluster:

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

   - Oracle JDK 8

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

   - Oracle JDK 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 command 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 Apache projects Hive and HCatalog, Oozie, Hue, or Ranger, 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.

Refer to the HDP 2.5.0 Release Notes for information on supported metastore databases.

1.1.5.1. Metastore Database Prerequisites

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

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

- For Hue: Hue user name and Hue user password
- For Apache Ranger: RANGER_ADMIN_DB_NAME

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. Installing PostgreSQL on RHEL, CentOS, and Oracle Linux

Use the following instructions to install a new instance of PostgreSQL:

1. Using a terminal window, connect to the host machine where you plan to deploy a PostgreSQL instance:

   ```
   yum install postgresql-server
   ```

2. Start the instance:

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

   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.

  Remove comments from the "port = " line and specify the port number (default 5432).

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

  the following:

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

- 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 by logging

   in as the root user and entering the following syntax:

   ```
   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
   ```

   The previous syntax should have the following values:

   - $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, you must create hive_dbuser after Hive has been installed, and for access to the Oozie metastore, you must 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` file is in the Java share directory:

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

1.1.5.2.2. Installing PostgreSQL on 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 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.

   ```
   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
   ```

1.1.5.2.3. Installing PostgreSQL on Ubuntu and Debian

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:

   ```
   apt-get install postgresql-server
   ```

2. Start the instance.

   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.

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

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

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 psql -U postgres
   echo "GRANT ALL PRIVILEGES ON DATABASE $dbname TO $user;" | sudo -u psql -U
    postgres
   ```

   Where:

   $postgres is the postgres user, $user is the user you want to create, and $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.

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. Installing MySQL on RHEL and 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 CentOS 6.6)

   `yum install mysql-community-release` For CentOS 7.x, 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, use the following command 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
   ```

   In this syntax, "root" is the root user password.

7. Log in as the root user, create the “dbuser,” and grant dbuser adequate privileges:

   ```
   [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" by 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 by using the following command format:

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

   For example, to set the password to "root", use the following command:

```
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 dbuser adequate privileges:

   ```
   [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 by 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*
```

1.1.5.3.3. Ubuntu/Debian

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:

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

3. Start the instance.

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

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

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

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

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

5. Remove unnecessary information from log and STDOUT.

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

6. Log in to MySQL as the root user:

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

7. Log in as the root user, 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.

    ```
    apt-get 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, and then follow the instructions in "Set up Oracle DB for use with Hive Metastore" in this guide.

1.2. Virtualization and Cloud Platforms

Hortonworks Data Platform (HDP) is certified and supported when running on virtual or cloud platforms (for example, VMware vSphere or Amazon Web Services EC2) if 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. Configuring 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. To download the HDP maven artifacts and build your own repository, see Download the HDP Maven Artifacts

Note

If your cluster does not have access to the Internet, or if 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/2.5.0.
0/hdp.repo
```

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

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos7/2.x/updates/2.5.0.
0/hdp.repo
```

- SLES

```
wget -nv http://public-repo-1.hortonworks.com/HDP/suse<version>sp<number>/2.
x/updates/2.5.0.0/hdp.repo
```

- Ubuntu

```
apt-get update
wget http://public-repo-1.hortonworks.com/HDP/ubuntu<version>/2.x/updates/2.
5.0.0/hdp.list -O /etc/apt/sources.list.d/hdp.list
```

- Debian

```
apt-get update
wget http://public-repo-1.hortonworks.com/HDP/debian7/2.x/updates/2.5.0.0/
hdp.list -O /etc/apt/sources.list.d/hdp.list
```

1.4. Deciding on a Deployment Type

While it is possible to deploy all of HDP on a single host, 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 Apache Hive, HCatalog, or Apache Oozie, you need the host name, database name, user name, 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 permissions on that instance.

1.6. Prepare the Environment

To deploy your HDP instance, you must prepare your deployment environment:

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

1.6.1. Enable NTP on Your Cluster

The clocks of all the nodes in your cluster must be synchronized. If your system does not have access to the Internet, you should set up a master node as an NTP xserver to achieve this synchronization.

Use the following instructions to enable NTP for your cluster:

1. Configure NTP clients by executing the following command on each node in your cluster:
   - For RHEL/CentOS/Oracle Linux 6.x

     ```
     yum install ntp
     ```

   - For RHEL/CentOS/Oracle Linux 7.x:

   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
     ```

   - For Ubuntu and Debian:

```
apt-get install ntp
```

2. Enable the service by executing the following command on each node in your cluster:

   - For RHEL/CentOS/Oracle Linux:

     ```
     chkconfig ntpd on
     ```

   - For SLES, Ubuntu, and Debian:

     ```
     chkconfig ntp 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
     ```

   - For Ubuntu and Debian:

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

4. If you want to use an existing NTP server as the X 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, as shown in the following example using RHEL hosts:

   ```
   # 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.

   - Disable SELinux temporarily by executing the following command:

     ```
     setenforce 0
     ```

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

1.6.3. Disable IPTables

Because certain ports must be open and available during installation, you should temporarily disable iptables. If the security protocols at your installation do not allow you to disable iptables, you can proceed with them on if all of the relevant ports are open and available; otherwise, cluster installation fails. 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 firewall:

  ```
  systemctl stop firewalld
  ```

  ```
  systemctl mask firewalld
  ```

  Restart firewalld after your setup is complete.

- On Ubuntu and Debian host machines, execute the following command to disable iptables:

  ```
  service ufw stop
  ```

  Restart iptables after your setup is complete.

Important

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

1.7. Download Companion Files

You can download and extract a set of companion files, including script files and configuration files, that you can then modify to match your own cluster environment:

To download and extract the files:

```
wget http://public-repo-1.hortonworks.com/HDP/tools/2.5.0.0/
hdp_manual_install_rpm_helper_files-2.5.0.0.1245.tar.gz
tar zxvf hdp_manual_install_rpm_helper_files-2.5.0.0.1245.tar.gz
```

1.8. Define Environment Parameters

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

1. Define directories.

   The following table describes the directories you need for installation, configuration, data storage, process IDs, and log information based on the Apache 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 that you downloaded in the supplied companion files includes a script, directories.sh, for setting directory environment parameters.

   You should edit and source (or copy the contents to your ~/.bash_profile) to set up these environment variables in your environment.

Table 1.1. Directories Needed to Install Core Hadoop

Hadoop Service	Parameter	Definition
HDFS	DFS_NAME_DIR	Space separated list of directories to which 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 stores 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. Directories Needed to Install Ecosystem Components

Hadoop Service	Parameter	Definition
Pig	PIG_CONF_DIR	Directory in which to store the Apache 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 Apache 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 Apache ZooKeeper stores 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 Apache Sqoop configuration files. For example, /etc/sqoop/conf.

If you use the companion files, the following screen 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 requires you to
# change values in other configuration files.
```

```
#

#
# Hadoop Service - HDFS
#

# Space separated list of directories where NameNode stores 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 stores 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 stores
 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 stores 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 stores 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 stores 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 Hadoop Distributed File Sysem (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 Apache Falcon services. For example, falcon.
SQOOP_USER	User owning the Sqoop services. For example, sqoop.
KAFKA_USER	User owning the Apache 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. Creating System Users and Groups

In general, Apache Hadoop services should be owned by specific users and not by root or application users. The following table shows the typical users for Hadoop services. If you choose to install the HDP components using the RPMs, these users are automatically 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 follow this procedure:

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

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

1.10. Determining HDP Memory Configuration Settings

You can use either of two methods determine YARN and MapReduce memory configuration settings:

- Running the HDP Utility Script
- Manually Calculating YARN and MapReduce Memory Configuration Settings

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

1.10.1. Running the YARN Utility Script

This section describes how to use the yarn-utils-py script to calculate YARN, MapReduce, Hive, and Tez memory allocation settings based on the node hardware specifications. The yarn-utils-py script is included in the HDP companion files. See Download Companion Files.

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

Table 1.5. yarn-utils-py Options

Option	Description
-c CORES	The number of cores on each host
-m MEMORY	The amount of memory on each host, in gigabytes
-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-2.5.0.0.1245 directory:

```
python yarn-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. Calculating YARN and MapReduce Memory Requirements

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 an Apache 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, you should 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 = stack memory reserve + HBase memory reserve (if HBase is on the same node)

You can use the values in the following table to determine what you need for 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	2 GB	2 GB
24 GB	4 GB	4 GB
48 GB	6 GB	8 GB
64 GB	8 GB	8 GB
72 GB	8 GB	8 GB
96 GB	12 GB	16 GB
128 GB	24 GB	24 GB
256 GB	32 GB	32 GB
512 GB	64 GB	64 GB

After you determine the amount of memory you need per node, you must determine the maximum number of containers allowed per node:

# of containers = min (2*CORES, 1.8*DISKS, (total available RAM) / MIN_CONTAINER_SIZE)

DISKS is the value for dfs.data.dirs (number of data disks) per machine.

MIN_CONTAINER_SIZE is the minimum container size (in RAM). This value depends on the amount of RAM available; in smaller memory nodes, the minimum container size should also be smaller.

The following table provides the recommended values:

Table 1.7. Recommended Container Size 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

Finally, you must determine the amount of RAM per container:

RAM per container = max(MIN_CONTAINER_SIZE, (total available RAM, per containers)

Using the results of all the previous calculations, you can configure YARN and MapReduce.

Table 1.8. YARN and MapReduce Configuration Values

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

Assume that your cluster nodes have 12 CPU cores, 48 GB RAM, and 12 disks:

Reserved memory = 6 GB system memory reserve + 8 GB for HBase min container size = 2 GB

If there is no HBase, then you can use the following calculation:

# 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 Without HBase

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 with HBase

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

Notes:

- 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 or cores, you can free 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 preconfigured 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 the previous example cluster, with the previous configurations, YARN is 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 in which 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. Recommended 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

Number of Files, in Millions	Total Java Heap (Xmx and Xms)	Young Generation Size (-XX:NewSize -XX:MaxNewSize)
20-30	14848m	2048m
30-40	19456m	2560m
40-50	24320m	3072m
50-70	33536m	4352m
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.

Following are the recommended settings for HADOOP_NAMENODE_OPTS in the hadoop-env.sh file (replacing 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:

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

1.12. Allocating 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. Downloading the HDP Maven Artifacts

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

Other than the release artifacts, some non-Hortonworks artifacts are necessary for building the HDP stack. These third-party artifacts are hosted in the Hortonworks nexus repository:

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

and

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

If developers want to develop an application against the HDP stack, and they also have a maven repository manager in-house, then they can proxy these three repositories and continue referring to the internal maven groups repo.

If developers do not have access to their in-house maven repos, they can directly use the Hortonworks public groups repo http://repo.hortonworks.com/content/groups/public/ and continue to develop 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.

1. Install the ZooKeeper Package
2. Securing Zookeeper with Kerberos
3. Securing Zookeeper Access
4. Set Directories and Permissions
5. Set Up the Configuration Files
6. Start Zookeeper

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 is 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
  ```

- For Ubuntu and Debian:

  ```
  apt-get install zookeeper
  ```

Note

Grant the zookeeper user shell access on Ubuntu and Debian.plevinson 5/9/2016 HDP

```
usermod -s /bin/bash 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
   ```

Note

Grant the zookeeper user shell access on Ubuntu and Debian.

```
usermod -s /bin/bash zookeeper
```

2.3. Securing ZooKeeper Access

The default value of `yarn.resourcemanager.zk-acl` allows anyone to have full access to the znode. Hortonworks recommends that you modify this permission to restrict access by performing the steps in the following sections.

- ZooKeeper Configuration

- YARN Configuration
- HDFS Configuration

2.3.1. ZooKeeper Configuration

Note

The steps in this section only need to be performed once for the HDP cluster. If this task has been done to secure HBase for example, then there is no need to repeat these ZooKeeper steps if the YARN cluster uses the same ZooKeeper server.

1. Create a keytab for ZooKeeper called `zookeeper.service.keytab` and save it to `/etc/security/keytabs`.

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

2. Add the following to the `zoo.cfg` file:

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

3. Create the `zookeeper_client_jaas.conf` file.

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

4. Create the `zookeeper_jaas.conf` file.

   ```

   Server {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   storeKey=true
   useTicketCache=false
   keyTab="$PATH_TO_ZOOKEEPER_KEYTAB"
   (such as"/etc/security/keytabs/zookeeper.service.keytab")
   principal="zookeeper/$HOST";
   (such as "zookeeper/xuan-sec-yarn-ha-2.novalocal@SCL42.HORTONWORKS.COM";)
   };
   ```

5. Add the following information to `zookeeper-env-sh`:

   ```
   export CLIENT_JVMFLAGS="-Djava.security.auth.login.config=/etc/zookeeper/
   conf/zookeeper_client_jaas.conf"
   export SERVER_JVMFLAGS="-Xmx1024m
   -Djava.security.auth.login.config=/etc/zookeeper/conf/zookeeper_jaas.conf"
   ```

2.3.2. YARN Configuration

Note

The following steps must be performed on all nodes that launch the ResourceManager.

1. Create a new configuration file called `yarn_jaas.conf` in the directory that contains the Hadoop Core configurations (typically, `/etc/hadoop/conf`).

```
Client {
com.sun.security.auth.module.Krb5LoginModule required
useKeyTab=true
storeKey=true
useTicketCache=false
keyTab="$PATH_TO_RM_KEYTAB"
(such as "/etc/security/keytabs/rm.service.keytab")
principal="rm/$HOST";
(such as "rm/xuan-sec-yarn-ha-1.novalocal@EXAMPLE.COM";)
};
```

2. Add a new property to the `yarn-site.xml` file.

```

<property>
<name>yarn.resourcemanager.zk-acl</name>
<value>sasl:rm:rwcda</value>
</property>
```

Note

Because `yarn-resourcemanager.zk-acl` is set to `sasl`, you do not need to set any value for `yarn.resourcemanager.zk-auth`. Setting the value to sasl also means that you cannot run the command `addauth<scheme><auth>` in the zkclient CLI.

3. Add a new YARN_OPTS to the `yarn-env.sh` file and make sure this YARN_OPTS is picked up when you start your ResourceManagers.

```
YARN_OPTS="$YARN_OPTS -Dzookeeper.sasl.client=true
-Dzookeeper.sasl.client.username=zookeeper
-Djava.security.auth.login.config=/etc/hadoop/conf/yarn_jaas.conf
-Dzookeeper.sasl.clientconfig=Client"
```

2.3.3. HDFS Configuration

1. In the `hdfs-site.xml` file, set the following property, for security of ZooKeeper based fail-over controller. when NameNode HA is enabled:

```
<property>
    <name>ha.zookeeper.acl</name>
    <value>sasl:nn:rwcda</value>
</property>
```

2.4. 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 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 stores 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**

2.5. Set Up the Configuration Files

You must 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 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.

5. 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
   ```

6. 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.

2.6. 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
2. Install the Hadoop packages
3. Install compression libraries
4. Create directories

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
  ```

- For Ubuntu/Debian:

  ```
  apt-get install hadoop hadoop-hdfs libhdfs0 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
  ```

- For Ubuntu/Debian:

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

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
  ```

- For Ubuntu/Debian:

  ```
  apt-get install liblzo2-2 liblzo2-dev hadooplzo
  ```

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 need to modify them to match your own cluster environment.

Use the following instructions to create appropriate directories:

1. Create the NameNode Directories

2. Create the SecondaryNameNode Directories
3. Create the DataNode and YARN NodeManager Local dDirectories
4. Create the Log and PID Directories
5. Symlink Directories with hdp-select

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 create directories for each service. If you choose to use the companion file scripts, these environment variables are already 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 2.5.0 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 2.5.0.0-<$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 2.5.0.0-<$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 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`:

```
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.

8. Set the Concurrent Mark-Sweep (CMS) Garbage Collector (GC) parameters.

   On the NameNode host, open the `etc/hadoop/conf/hadoop-env.sh` file. Locate `export HADOOP_NAMENODE_OPTS=<parameters>` and add the following parameters:

   ```
   -XX:+UseCMSInitiatingOccupancyOnly
   -XX:CMSInitiatingOccupancyFraction=70
   ```

By default CMS GC uses a set of heuristic rules to trigger garbage collection. This makes garbage collection less predictable and tends to delay collection until the old generation is almost fully occupied. Initiating it in advance allows garbage collection to complete before the old generation is full, and thus avoid Full GC (i.e. a stop-the-world pause).

- `-XX:+UseCMSInitiatingOccupancyOnly` prevents the use of GC heuristics.
- `-XX:CMSInitiatingOccupancyFraction=<percent>` tells the Java VM when CMS should be triggered. Basically, it allows the creation of a buffer in heap, which can be filled with data while CMS is running. This percent should be back-calculated from the speed with which memory is consumed in the old generation during production load. If this percent is set too low, the CMS runs too often; if it is set too high, the CMS is triggered too late andconcurrent mode failure may occur. The recommended setting for `-XX:CMSInitiatingOccupancyFraction` is 70, which means that the application should utilize less than 70% of the old generation.

5. Validating the Core Hadoop Installation

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

1. Format and start HDFS
2. Smoke test HDFS
3. Configure YARN and MapReduce
4. Start YARN
5. Start the MapReduce JobHistory Server
6. Smoke test MapReduce

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 2.5.0.0.

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 returns 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 -chmod -R 1777 /mr-history/tmp

hdfs dfs -mkdir -p /mr-history/done
hdfs dfs -chmod -R 1777 /mr-history/done
hdfs dfs -chown -R $MAPRED_USER:$HDFS_USER /mr-history

hdfs dis -mkdir -p /app-logs
hdfs dfs -chmod -R 1777 /app-logs

hdfs dfs -chown $YARN_USER:$HDFS_USER/app-logs
```

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.

1. Install the HBase Package
2. Set Directories and Permissions
3. Set up the Configuration Files
4. Validate the Installation
5. Start the HBase Thrift and REST Servers

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
```

- For Ubuntu

```
apt-get 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 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

Note

To enable Kerberos for clusters with dual home network setting, each HBase RegionServer must have its own key. See Setting Up Security for Manual Installs.

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 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</name>
            <value>kerberos</value>
            <description^>Enter the authentication method for the REST server.
       </description>
       </property>

       <property>
            <name>hbase.rest.kerberos.spnego.principal</name>
            <value>HTTP/_HOST@EXAMPLE.COM</value>
            <description>Enter the hostname of the principal.</description>
       </property>

       <property>
            <name>hbase.rest.keytab.file</name>
            <value>/etc/security/keytabs/spnego.service.keytab</value>
            <description>Enter the location of the keytab.</description>
       </property>
       ```

  - After adding these properties to hbase-site.xml, you must grant HBase permissions to the user:

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

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

    ```
    grant 'HTTP', 'RWXCA', 'sales', '1'
    ```

- 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. Starting 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 RegionServers and all master nodes.

1. Installing the Phoenix Phoenix
2. Configuring HBase for Phoenix
3. Configuring Phoenix to Run in a Secure Cluster
4. Validating the Phoenix Installation
5. Troubleshooting Phoenix

7.1. Installing the Phoenix Package

Run the following command to install Phoenix:

- RHEL/CentOS/Oracle Linux

  ```
  yum install phoenix
  ```

- SLES

  ```
  zypper install phoenix
  ```

- Ubuntu

  ```
  apt-get 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 RegionServers.

   - 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. The property settings also ensure 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 the `hbase-site.xml` file on all HBase nodes.

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

3. Restart the HBase Master server and the RegionServers.

Note

Repairing overlap regions inconsistencies using the **hbck** tool can result in a situation where local indexes are inconsistent with the data in the table. If you know the database schema, you can fix this issue by dropping and recreating all local indexes of the table after the **hbck** tool completes its operation. Alternatively, you can rebuild the local indexes using the following ALTER query:

```
ALTER INDEX IF EXISTS index_name ON data_table_name REBUILD
```

Note

During local index creation manual splits and bulk loads fail because they are disabled. Avoid massive data insertion (bulk loads and batch upserts) during local index creation.

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.

1. Prerequisites
2. Installing the Tez Package
3. Configuring Tez
4. Creating a New Tez Instance
5. Validating the Tez Installation
6. Troubleshooting

8.1. Prerequisites

Verify that your cluster is upgraded to HDP 2.5.0 or higher before installing Tez.

Note

Please note that the instructions for installing and configuring Tez on HDP 2.5.0 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
     ```

   - For Ubuntu or Debian:

     ```
     apt-get 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 2.5.0.0.

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/2.5.0.0-1245/
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 2.5.0.0.

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	20000

Configuration Parameter	Description	Default Value
	assigned to it immediately. Only active when reuse is enabled.	
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
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 are 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.	INFO

Configuration Parameter	Description	Default Value
	`org.apache.hadoop.ipc=INFO;org.apache.hadoop.security=INFO` is *not* valid.	
tez.am.max.app.attempts	Specifies the total number of times that the app master is re-run in case recovery is triggered.	2
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 are 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 is 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	0.8

Configuration Parameter	Description	Default Value
	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 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 ${fs.defaultFS}.	org.apache.tez.dag.history.logging.ats.ATSHistoryLoggingService
tez.lib.uris	Comma-delimited list of the location of the Tez libraries which is localized for DAGs. Specifying a single .tar.gz or .tgz assumes that a compressed version of the tez libs is being used. This is uncompressed into a tezlibs directory when running containers, and tezlibs/;tezlibs/lib/ are added to the classpath (after . and .*). If multiple files are specified - files are localized as regular files, contents of directories are localized as regular files (non-recursive).	There is no default value, but it should be set to: `/hdp/apps/${hdp.version}/tez/tez.tar.gz`, or the location of the `tez` tarball on HDFS, or the appropriate distributed filesystem path.
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

Configuration Parameter	Description	Default Value
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
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 is not blocked 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 renames CounterGroups/CounterNames, making them unique per vertex edge instead of unique per vertex.	true

Configuration Parameter	Description	Default Value
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
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 are 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

Configuration Parameter	Description	Default Value
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
tez.use.cluster.hadoop-libs	Specifies whether Tez uses 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:

1. Installing the Hive-HCatalog Package
2. Setting Directories and Permissions
3. Setting Up the Hive/HCatalog Configuration Files
4. Setting Up the Database for the Hive Metastore
5. Setting up RDBMS for use with Hive Metastore
6. Creating Directories on HDFS
7. Enabling Tez for Hive Queries
8. Disabling Tez for Hive Queries
9. Configuring Tez with the Capacity Scheduler
10. Validating Hive-on-Tez Installation

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 are executed), Hive Metastore Server, and HiveServer2 machine, install the Hive RPMs.

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install hive-hcatalog
  ```

- For SLES:

  ```
  zypper install hive-hcatalog
  ```

- For Ubuntu:

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

2. **(Optional**) Download and 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 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*
        ```

      - For UBUNTU/Debian:

        ```
        apt-get 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 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 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. The datanucleus.cache.level2 configuration parameter is ignored, and assigning a value of none to this parameter does 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 Configuration 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 properties 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 are executed using Tez or MapReduce.	If this value is set to "mr," Hive queries are executed using MapReduce. If this value is set to "tez," Hive queries are executed using Tez. All queries executed through HiveServer2 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) are 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) are 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 are 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 are 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 is 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 an Oracle database 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 administration 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/warehouse
   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. Run the following command to copy the `hive-exec-*.jar` to HDFS at `/apps/hive/install/`:

```
su - $HIVE_USER
hadoop fs -mkdir /apps/hive/install
hadoop fs -copyFromLocal /usr/hdp/<hdp_version>/hive/lib/hive-exec-*<hdp
 version>*.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 is then disabled for Hive queries.

9.9. Configuring Tez with the Capacity Scheduler

You can use the tez.queue.name property to specify which queue is 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
```

9.11. Installing Hive LLAP

Note

This 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 technical support by logging a case on the Hortonworks Technical Support Portal.Hortonworks Technical Support Portal

LLAP provides you with extended query execution in Apache Hive that consists of:

- A persistent daemon that provides execution and caching of data
- A tightly integrated directed acyclic graph (DAG)-based framework

The primary benefit of using LLAP is to enhance your query execution times. Small, short queries can be processed by the daemon. For more information about LLAP, refer to

9.12. Prerequisites

Before installing LLAP, make sure that your cluster meets the following prerequisites:

Table 9.2.

Prerequisite	Description
Cluster Stack Version	HDP 2.5.0
(Optional) Ambari Version	Apache Ambari 2.4.0
Components	• Apache HDFS • Apache Hive • Apache MapReduce2 • Apache Slider • Apache Tez • Apache YARN • Apache ZooKeeper
Configuration Files	• tez-site.xml • hive-site.xml
JDK	JDK8u60 or later

9.13. Preparing to Install LLAP

Note

HDP 2.5 ships with Hive and and Hive 2.0, and Tez and Tez_hive2. Install Hive and Tez as usual; install Hive 2.0 and Tez_hive2 on the boxes where HiveServer2 will run and where the Hive client exists.

1. Install Java JDK8u60 or later to the same path on each server in your cluster.
2. Set the JAVA_HOME environment variable to the path where you install the JDK and then export it to child processes:

```
export JAVA_HOME=<path/to/jdk1.8.0_60>
export PATH=$JAVA_HOME/bin:$PATH:
```

9.14. Installing LLAP on an Unsecured Cluster

On the computer where Hive2 is installed, follow these steps:

1. Log in as the Hive user:

```

sudo su -
su $HIVE_USER
```

2. Generate the LLAP package:

```
hive --service llap --name <llap_svc_name> --instances
<number_of_cluster_nodes>
--cache <cache_size>m --xmx <heap_size>m --size
((<cache_size>+<heap_size>)*1.05)m
--executors <number_of_cores> --loglevel <WARN|INFO>
--args " -XX:+UseG1GC -XX:+ResizeTLAB -XX:+UseNUMA  -XX:-ResizePLAB"
```

Table 9.3.

Parameter	Recommended Value
–instances <cache_size>	Number of cluster nodes to use LLAP.
–cache <cache_size>	<YARN_maximum_container_size> - (<hive.tez.container.size> <number_of_cores>) <hive.tez.container.size> is the setting for this property, found in the `hive-site.xml` file. Depending on the size of the node, specify a minimum between 1 GB through 4 GB.
–xmx <heap_size>	For medium-sized nodes: <hive.tex.container.size> * <number of cores> * (0.8 to 0.95) <hive.tez.container.size> is the setting for this property found in the `hive-site.xml` file. Ensure that the setting for –xmx is 1 GB less than (<hive.tez.container.size> * <number_of_cores>). For smaller nodes, use the same formula, multiplied by 0.8.
–executors <number_of_cores>	Set to the number of CPU cores available on nodes running NodeManager. Set this value even if CPU scheduling is enabled in YARN.

3. Set the `--loglevel` parameter to `INFO` when you are troubleshooting or testing.

   The `INFO` option provides verbose output. In a production environment, you would set the `--loglevel` parameter to WARN, which outputs a message to the logs only if there is a warning or error. This makes the logs easier to read and reduces load on the node.

   ### Note

   The recommended values listed in the previous table represent a sample configuration. LLAP also can be configured to use a fraction of node resources.

   The following message is returned, confirming that the LLAP package was created: `Prepared llap-slider-<date>/run.sh for running LLAP on Slider`

4. Verify that the following LLAP and HiveServer2 properties are set in the `hive-site.xml` file, which can be located in the `/etc/hive/conf/` or the `/etc/hive/conf/conf.server/` directory:

Table 9.4. LLAP Properties to Set in hive-site.xml

Property	Values
`hive.execution.mode`	llap
hive.llap.execution.mode	all
hive.llap.daemon.service.hosts	@<llap_service_name>

<llap_service_name> is the service name that you specified when you generated the LLAP package in Step 2.

Table 9.5. HiveServer2 Properties to Set in hive-site.xml to Enable Concurrent Queries with LLAP

Property	Values
hive.server2.tez.default.queues	Queue to use for the LLAP sessions. These correspond to YARN queues of the same name. For example, default.
hive.server2.tez.initialize.default.sessions	True
hive.server2.tez.sessions.per.default.queue	Set to the number of concurrent queries to run on the queues that are configured byhive.server2.tez.default.queues. This setting launches long-running Tez AMs (sessions).

5. Example of these properties set in the `hive-site.xml` file:

```
<property>
     <name>hive.execution.mode</name>
     <value>llap</value>
</property>

<property>
     <name>hive.llap.execution.mode</name>
     <value>ALL</value>
</property>
```

```
<property>
     <name>hive.llap.daemon.service.hosts</name>
     <value>@<llap_service_name></value>
</property>

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

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

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

9.15. Installing LLAP on a Secured Cluster

9.15.1. Prerequisites

- Cluster is available and is already secured with Kerberos.
- Slider and ZooKeeper are installed on the cluster.
- The Hadoop directory is in the same location on each node so the native binary can be accessed, which supports secure I/O.

Important

- You should have a method to execute commands and upload files on all nodes in the cluster from one "setup" or "jump" node. This can be set up with passwordless ssh (pssh) or by using a FOR loop with a list of nodes.
- On the "setup" node, add Java tools to the system path.
- All passwords, user names, and paths in this section of the document are provided as examples unless otherwise noted. Change them for your deployment as appropriate.

9.15.2. Installing LLAP on a Secured Cluster

Review the prerequisites for installing LLAP on a secured cluster before you begin.

1. Ensure that the Hive user exists on each node, and configure the following:

   a. Create local directories that are similar to those set up for the yarn.nodemanager.local-dirs property:

   ```
   mkdir -p /grid/0/hadoop/llap/local
   chown -R hive /grid/0/hadoop/llap
   ```

   b. On the "setup" node, ensure that the Hive user has access to its HDFS home directory:

```
hadoop fs -mkdir -p /user/hive
hadoop fs -chown -R hive /user/hive r
```

2. Set up keytabs.

   You can perform this step on the "setup" machine and distribute it to the cluster, or you can perform this step on each node. The following example shows how to perform this step on each node. Use kadmin.local if under root; otherwise, use kadmin.

3. On each node (specified by their fully qualified domain names), create the host and headless principals, and a keytab with each:

```
kadmin.local -q 'addprinc -randkey hive@EXAMPLE.COM'
 kadmin.local -q "addprinc -randkey hive/<fqdn>@EXAMPLE.COM"
 kadmin.local -q 'cpw -pw hive hive'
 kadmin.local -q "xst -norandkey -k /etc/security/keytabs/hive.keytab hive/
<fqdn>@EXAMPLE.COM"
kadmin.local -q "xst -norandkey -k /etc/security/keytabs/hive.keytab
 hive@EXAMPLE.COM"
chown hive /etc/security/keytabs/hive.keytab
```

4. On the "setup" node, create and install, as the Hive user, the headless keytab for Slider:

```
kadmin.local -q "xst -norandkey -k hive.headless.keytab hive@EXAMPLE.COM"
chown hive hive.headless.keytab
kinit -kt /etc/security/keytabs/hive.keytab hive@EXAMPLE.COM
slider install-keytab --keytab hive.headless.keytab --folder hive --
overwrite
```

5. If you want to use web UI SSL, set up the keystore for SSL.

   Note that Keystore is often set up for other web UIs: for example HiveServer2. If the keystore is not already set up, perform the following steps:

   a. Create the Certificate Authority (CA).

      i. On the setup node, create the CA parameters:

```
cat > /tmp/cainput << EOF
US
California
Palo Alto
Example Certificate Authority
Certificate Authority
example.com
.
EOF
```

      ii. Create the CA:

Note

The JAVA_HOME must be set. The default Java truststore password must be changed.

```
mkdir -p /etc/security/certs/
openssl genrsa -out /etc/security/certs/ca.key 4096
```

```
cat /tmp/cainput | openssl req -new -x509 -days 36525 -key /etc/
security/certs/ca.key \
  -out /etc/security/certs/ca.crt
echo 01 > /etc/security/certs/ca.srl
echo 01 > /etc/security/certs/ca.ser
keytool -importcert -noprompt -alias example-ca –keystore \
  $JAVA_HOME/jre/lib/security/cacerts -storepass changeit -file \
  /etc/security/certs/ca.crt
rm /tmp/cainput
```

b. Create the certificate.

   On the "setup" node, create the keystore parameters. In the following example, llap00 is the password specified for the new keystore:

```
hostname -f > /tmp/keyinput
hostname -d >> /tmp/keyinput
cat >> /tmp/keyinput << EOF
Example Corp
Palo Alto
CA
US
yes
llap00
llap00
EOF
```

c. Generate a keystore, a certificate request, and a certificate, and then import the certificate into the keystore:

```
cat /tmp/keyinput | keytool -genkey -alias hive -keyalg RSA -keystore \
  /etc/security/certs/keystore.jks -keysize 4096 -validity 36525 -
storepass llap00
keytool -certreq -alias hive -keystore /etc/security/certs/keystore.jks \
  -storepass llap00 -file /etc/security/certs/server.csr
openssl x509 -req -days 36525 -in /etc/security/certs/server.csr \
 -CA /etc/security/certs/ca.crt -CAkey /etc/security/certs/ca.key \
 -CAserial /etc/security/certs/ca.ser -out /etc/security/certs/server.crt
keytool -import -alias hive -keystore /etc/security/certs/keystore.jks \
  -storepass llap00 -trustcacerts -file /etc/security/certs/server.crt
chown hive:hadoop /etc/security/certs/keystore.jks /etc/security/certs/
server.crt
chmod 640 /etc/security/certs/keystore.jks /etc/security/certs/server.crt
rm /tmp/keyinput
```

d. Distribute the keystore and certificate to each node:

   i. On each node, create the directory:

```
mkdir -p /etc/security/certs
```

   ii. Upload the files from the "setup" node:

```
scp … /etc/security/certs/* …@node:/etc/security/certs/
```

   iii. Import the CA:

```
chown hive:hadoop /etc/security/certs/*
```

```
chmod 640 /etc/security/certs/keystore.jks /etc/security/certs/server.
crt
keytool -importcert -noprompt -alias example-ca -keystore \
  $JAVA_HOME/jre/lib/security/cacerts -storepass changeit -file \
  /etc/security/certs/ca.crt
```

e. Configure LLAP and generate the package.

Specify the following properties in the `/etc/hive/conf/hive-site.xml` file:

Table 9.6. Properties to Set in hive-site.xml for Secured Clusters

Property	Values
hive.llap.daemon.work.dirs	hive.llap.daemon.work.dirs
hive.llap.daemon.keytab.file	hive.llap.daemon.keytab.file
hive.llap.daemon.service.principal	hive.llap.daemon.service.principal
hive.llap.daemon.service.ssl	True
hive.llap.zk.sm.principal	hive@EXAMPLE.COM
hive.llap.zk.sm.keytab.file	/etc/security/keytabs/hive.keytab
hive.llap.zk.sm.connectionString	ZooKeeper connection string: for example, <machine:port,machine:port, ...>
hadoop.security.authentication	kerberos
hadoop.security.authorization	true

Following is an example of these properties set in the llap-daemon-site.xml file:

```
<property>
     <name>hive.llap.daemon.work.dirs</name>
     <value>/grid/0/hadoop/llap/local</value>
</property>

<property>
     <name>hive.llap.daemon.keytab.file</name>
     <value>/etc/security/keytabs/hive.keytab</value>
</property>

<property>
     <name>hive.llap.daemon.service.principal</name>
     <value>hive/_HOST@EXAMPLE.COM</value>
</property>

<property>
     <name>hive.llap.daemon.service.ssl</name>
     <value>true</value>
</property>

<property>
     <name>hive.llap.zk.sm.principal</name>
     <value>hive@EXAMPLE.COM</value>
</property>

<property>
     <name>hive.llap.zk.sm.keytab.file</name>
     <value>>/etc/security/keytabs/hive.keytab</value>
</property>

<property>
```

```
     <name>hive.llap.zk.sm.connectionString</name>
     <value>127.0.0.1:2181,128.0.0.1:2181,129.0.0.1:2181</value>
</property>

<property>
     <name>hadoop.security.authentication</name>
     <value>kerberos</value>
</property>

<property>
     <name>hadoop.security.authorization</name>
     <value>true</value>
</property>
```

Optionally, you can also use hive.llap.daemon.acl and hive.llap.management.acl to restrict access to the LLAP daemon protocols.

The Hive user must have access to both.

a. Specify the following properties in the `ssl-server.xml` file.

   Ensure that you perform this step before you create the LLAP package.

Table 9.7. Properties to Set in ssl-server.xml for LLAP on Secured Clusters

Property	Values
ssl.server.truststore.location	Path to Java truststore: for example, /jre/lib/security/cacerts
ssl.server.keystore.location	/etc/security/certs/keystore.jks
ssl.server.truststore.password	changeit Note: This is the default password.
ssl.server.keystore.password	llap00
ssl.server.keystore.keypassword	llap00

Following is an example of these properties set in the `ssl-server.xml` file:

```
<property>
     <name>ssl.server.truststore.location</name>
     <value>/jre/lib/security/cacerts</value>
</property>

<property>
     <name>ssl.server.keystore.location</name>
     <value>/etc/security/certs/keystore.jks</value>
</property>

<property>
     <name>ssl.server.truststore.password</name>
     <value>strong_password</value>
     </property>

<property>
     <name>ssl.server.keystore.password</name>
     <value>llap00</value>
</property>

<property>
     <name>ssl.server.keystore.keypassword</name>
     <value>llap00</value>
</property>
```

Generate the LLAP package.

Important

Ensure that JAVA_HOME and HADOOP_HOME are set before generating the package. JAVA_HOME and the site.global.library_path property in the appConfig.json configuration file are set using JAVA_HOME and HADOOP_HOME. If you see problems such as a missing native library, check the appConfig.json configuration file.

Make sure that LLAP package generation is done under the hive user because some HDFS paths in the configuration are user-specific. You can modify the paths after package generation

b. To generate the LLAP package, run the following command, setting parameters as described in the LLAP Package Parameters table:

```
hive --service llap --name <llap_svc_name> --instances
 <number_of_cluster_nodes>
--cache <cache_size>m --xmx <heap_size>m --size ((<cache_size>
+<heap_size>)*1.05)m
--executors <number_of_cores> --loglevel <WARN|INFO>
--args " -XX:+UseG1GC -XX:+ResizeTLAB -XX:+UseNUMA  -XX:-ResizePLAB"
```

Table 9.8. LLAP Package Parameters

Parameter	Recomended Value (based on daemons using all available node resources)
–instances <cache_size>	Set to number of cluster nodes that you to use for LLAP.
–cache <cache_size>	<YARN_maximum_container_size> - (<hive.tez.container.size> * <number_of_cores>) <hive.tez.container.size> is the setting for this property found in the hive-site.xml file. Depending on the size of the node, specify a minimum of 1-4 GB.
–xmx <heap_size>	For medium-sized nodes: <hive.tez.container.size> * <number_of_cores> * (0.8 to 0.95) Where <hive.tez.container.size> is the setting for this property found in the hive-site.xml file. Ensure that the setting for –xmx is 1GB less than (<hive.tez.container.size> * <number_of_cores>). For smaller nodes: Use the same formula as for medium-sized nodes, but multiply by 0.8
–executors <number_of_cores>	Set to the number of CPU cores available on nodes running NodeManager. Set this value even if CPU scheduling is enabled in YARN.

Set the –loglevel parameter to INFO when you are troubleshooting or testing. The INFO option provides verbose output. In a production environment, set the –loglevel parameter to WARN, which only outputs a message to the logs if there is a warning or error. This makes the logs easier to read and reduces load on the node.

Note

The recommended values listed in the LLAP Package Parameters table represent a sample configuration. LLAP also can be configured to use a fraction of node resources.

c. Specify the keytab settings in the slider-appmaster section of the appConfig.json configuration file if they have not already been specified:

```
"components": {
    "slider-appmaster": {
… existing settings …
"slider.hdfs.keytab.dir": ".slider/keytabs/llap",
"slider.am.login.keytab.name": "hive.headless.keytab",
"slider.keytab.principal.name": "hive@EXAMPLE.COM"
```

9.15.3. Validating the Installation on a Secured Cluster

1. Make sure that you are logged in as the Hive user.

2. Verify that the following properties are set as follows in the hive-site.xml file that is being used by HiveServer2:

   - hive.execution.mode = llap

   - hive.llap.execution.mode = all

   - hive.llap.daemon.service.hosts = @<llap_service_name>

3. From the hive user home directory, start the LLAP service:

   ```
   cd ~ ./llap-slider-<date>/run.sh
   ```

   <date> is the date that you generated the LLAP package. To verify that you have the correct <date>, on the node where you generated the LLAP package, make sure you are in the hive user home directory and view the subdirectories:

   ```
   cd ~ ls
   ```

4. There is a subdirectory named llap-slider-<date>. This subdirectory contains the run.sh script you use to start the LLAP service.

5. As user hive, run the Hive CLI and HiveServer2 to run test queries.

   If you are using the Hive CLI, you must kinit.

6. After running test queries, check the following:

   - Check the logs on YARN for the Slider application that is running LLAP.

Look for changes that indicate that LLAP is processing the test queries.

- Using the ResourceManager UI, monitor the Tez AM (session) to make sure that it does not launch new containers to run the query.

9.16. Stopping the LLAP Service

Where <llap_svc_name> is the name you specified when you generated the LLAP package, use the following command to stop the LLAP service:

```
slider stop <llap_svc_name>
```

9.17. Tuning LLAP for Performance

To make sure that the LLAP service is performing as well as possible, you might need to tune the following properties in the `/etc/hadoop/conf/yarn-site.xml` file:

- yarn.nodemanager.resource.memory-mb
- yarn.scheduler.minimum-allocation-mb
- yarn.scheduler.maximum-allocation-mb

For information about tuning these properties, see Determine HDP Memory Configuration Settings in this guide.

You can also tune performance for the LLAP service by monitoring the hive.tez.container.size property in the hive-site.xml configuration file. For more information about tuning this property, see the Hive Performance Tuning guide.

Finally, you can tune LLAP for performance by adding nodes to the cluster, which adds additional YARN NodeManagers.

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
2. Validate the Installation

10.1. Install the Pig Package

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

- For RHEL or CentOS:

  ```
  yum install pig
  ```

- For SLES:

  ```
  zypper install pig
  ```

- For Ubuntu or Debian:

  ```
  apt-get install pig
  ```

The package installs 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:

1. Install the WebHCat Package
2. Upload the Pig, Hive and Sqoop tarballs to HDFS
3. Set directories and permissions
4. Modify WebHCat configuration files
5. Set up the HDFS user and prepare the WebHCat directories
6. Validate the installation

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 pig
  ```

- For SLES:

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

- For Ubuntu:

  ```
  apt-get install hive-hcatalog hive-webhcat 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 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 are impersonated.

Variable	Value
hadoop.proxyuser.USER.hosts	A comma-separated list of the hosts that 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 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
2. Set Directories and Permission
3. Set Up the Oozie Configuration Files
4. Configure Your Database for Oozie
5. Set Up the Sharelib
6. Validate the Installation

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
     ```

- For Ubuntu and Debian:

  ```
  apt-get 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/
        ```

      - For Ubuntu and Debian:

        ```
        apt-get install extjs
        ```

        ```
        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. See Install Compression Libraries.

      ```
      ln /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
     ```

     ```
     ln /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. Update `/etc/oozie/conf/oozie-env.sh` with your environment information.

   Change the OOZIE_BASE_URL property to reflect your Oozie server hostname and port:

   ```
   export OOZIE_BASE_URL="http://<your Oozie Server Hostname>:<your
    OOZIE_HTTP_PORT>/oozie"
   ```

   For secure clusters, also add the following two lines that specify the jks path and a password to match the system configuration:

   ```
   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/2.5.0.0-1245/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/2.5.0.0-1245/oozie/tomcat-deployment/conf/ssl`, copy over that SSL directory to `usr/hdp/current/oozie-server/conf`:

```
cp -r /grid/0/hdp/2.5.0.0-1245/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 need 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 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. If you have custom Oozie actions, you must define them in `/etc/oozie/conf/oozie-site.xml`.

   - Edit the `/etc/oozie/conf/oozie-site.xml` file and add the following properties:

     ```
     <property>
     <name>oozie.service.SchemaService.wf.ext.schemas</name>
     <value>[Comma seperated list of custom actions]</value>
     </property>
     ```

     For example, if you have added Spark Action, enter the following:

```
<property>
<name>oozie.service.SchemaService.wf.ext.schemas</name>
<value>spark-action-0.1.xsd</value>
</property>
```

- 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>
  ```

- Add the following property for Spark action configuration:

  ```
  <property>
  <name>oozie.service.SparkConfigurationService.spark.configuration</name>
  <value>*=/etc/spark/conf/</value>
  </property>
  ```

  Addition of this property requires that you to restart the Oozie server.

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=${OOZIE_CONFIG:-/usr/hdp/2.5.0.0-1245/oozie/conf}
export OOZIE_DATA=${OOZIE_DATA:-/var/db/oozie}
export OOZIE_LOG=${OOZIE_LOG:-/var/log/oozie}
export CATALINA_BASE=${CATALINA_BASE:-/usr/hdp/2.5.0.0-1245/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. Set 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
   ```

2. Add the two zip files from spark-client python to spark share lib:

   a.
   ```
   hdfs dfs -put /usr/hdp/current/spark-client/python/lib/py4j-0.9-src.zip /
   user/oozie/share/lib/lib_<timestamp>/spark
   ```

   b.
   ```
   hdfs dfs -put /usr/hdp/current/spark–client/python/lib/pyspark.zip /user/
   oozie/share/lib/lib_<timestamp>/spark
   ```

   c. Update the oozie share lib:

   ```
   oozie admin –oozie http://oozie_server:11000/oozie -sharelibupdate
   ```

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.

12.7. Stop and Start Oozie

Use the `/usr/hdp/current/oozie-server/bin/oozied.sh` script with the *`start`* parameter to start the Oozie server.

Use the `/usr/hdp/current/oozie-server/bin/oozied.sh` script with the *`stop`* parameter to stop the Oozie server.

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 authorization, auditing, and data protection.

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

- Installation Prerequisites
- Installing Policy Manager
- Installing UserSync
- Installing Ranger Plug-ins
- Installing and Configuring Tag-Based Access Control
- Verifying the Installation

For information regarding Apache Ranger, refer to the HDP Security Features of the *Hadoop Security Guide*.

For information about installing Ranger using Ambari, see Installing Ranger Using Ambari.

13.1. Installation Prerequisites

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

- Solr 5.2 or above
- Refer to the HDP 2.5.0 Release Notes for information on supported JDKs.
- Refer to the HDP 2.5.0 Release Notes for information on supported operating systems.
- Refer to the HDP 2.5.0 Release Notes for information on supported metastore databases.

  If the database server is not installed at the same host, Ranger services need to have access to the database server host.
- If you plan to setup Ranger authorization to use LDAP/AD, refer to the Setting Up Hadoop Group Mapping for LDAP/AD in the *Hadoop Security Guide*.

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 2.5.0 resource-based service is added to your site's list of available repositories.

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

   - For RHEL6/Centos6/Oracle LINUX 6:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.5.0.0/
     hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - For Ubuntu

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDP/
     ubuntu<version>/2.x/GA/2.5.0.0/hdp.list -O /etc/apt/sources.list.d/hdp.
     list
     ```

   - For Debian

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDP/debian7/2.x/
     GA/2.5.0.0/hdp.list -O /etc/apt/sources.list.d/hdp.list
     ```

2. Find the Ranger Policy Admin software:

   a. For RHEL/Centos/Oracle LINUX:

      yum search ranger

   b. For Ubuntu, Debian:

      ```
      aptitude search ranger
      ```

3. Install the Ranger Policy Admin software:

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

4.
   ```
   apt-get install <package_name>
   ```

   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	/usr/share/java/mysql-connector-java.jar (default)	Y

Configuration Property	Default/Example Value	Required?
absolute path including the jar name.	/usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	
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		
audit_solr_urls	http://<solr_host>:8886/solr/ ranger_audits	Y
audit_solr_user		Y
audit_solr_password		Y
audit_solr_zookeepers		Only required if SolrCloud is used.
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)	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 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	none (default)	Y

Configuration Property	Default/Example Value	Required?
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 that 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.

- xa_ldap_userSearchFilter=(uid={0}) property at Ranger-admin side

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

In this section:

- Using the LDAP Connection Check Tool
- Install UserSync and Start the Service

13.3.1. Using the LDAP Connection Check Tool

The LDAP Connection Check tool is a command line tool that helps Ranger administrators configure LDAP properties for the UserSync module. The tool collects minimal input from the administrator about the LDAP/AD server and discovers various properties for users and groups in order to successfully pull only targeted Users and Groups from the LDAP/AD server. It provides options such as discovering/verifying UserSync-related properties as well as authentication properties, generating install properties for manual installation, etc. Once all of the required properties have been discovered and tested, these properties can be applied to the Ranger configuration during Ambari or non-Ambari cluster installation.

The LDAP Connection tool can be accessed in the `/usr/hdp/current/ranger-usersync/ldaptool` directory.

13.3.1.1. LDAP Connection Check Tool Parameters

You can use the `./run.sh -h` command to list the LDAP Connection Check tool parameters:

```
cd /usr/hdp/current/ranger-usersync/ldaptool
./run.sh -h
usage: run.sh
 -noauth         ignore authentication properties
 -d <arg>   {all|users|groups}
 -h                       show help.
 -i <arg>            Input file name
 -o <arg>            Output directory
 -r <arg>             {all|users|groups}
```

All these parameters are optional.

- If “-i” (for input file) is not specified, the tool will fall back to the CLI option for collecting values for mandatory properties.
- if “-o” (for output directory) is not specified, the tool will write all of the output files to the `/usr/hdp/current/ranger-usersync/ldaptool/output` directory.
- if “-noauth” (for ignoring authentication) is not specified, the tool will discovery and verify authentication-related properties.
- if “-d” (for discovering usersync properties) is not specified, the tool will default to discovering all of the usersync-related properties.
- if “-r” (for retrieving users and/or groups) is not specified, the tool will fallback to the “-d” option.

13.3.1.2. Input Properties

In order to discover the usersync and authentication related properties, the LDAP Connection Check tool collects some mandatory information as part of the input properties. These mandatory properties include:

- ranger.usersync.ldap.url (<ldap or ldaps>://<server ip/fqdn>:<port>)
- ranger.usersync.ldap.binddn (ldap user like AD user or ldap admin user)
- ranger.usersync.ldap.bindpassword (user password or ldap admin password)
- ranger.usersync.ldap.user.searchbase (Mandatory only for non AD environment)
- ranger.usersync.ldap.user.searchfilter (Mandatory only for non AD environment)
- ranger.admin.auth.sampleuser (Mandatory only for discovering authentication properties)

- ranger.admin.auth.samplepassword (Mandatory only for discovering authentication properties)

1. Modify the input.properties file provided as part of the tool installation and provide that file (with the complete path as the command line argument while running the tool.
2. Use the CLI to input the values for these mandatory properties.

The CLI option is provided to the user when the input file is not provided as the command line option (-i <arg>) while running the tool. Once the values are collected from the CLI, these values are stored in the input.properties file (in the conf dir of the installation folder) for later use.

The following is the CLI provided by the tool when input file is not specified. The tool provides two options for collecting values for these mandatory properties:

```
Ldap url [ldap://ldap.example.com:389]:
Bind DN [cn=admin,ou=users,dc=example,dc=com]:
Bind Password:
User Search Base [ou=users,dc=example,dc=com]:
User Search Filter [cn=user1]:
Sample Authentication User [user1]:
Sample Authentication Password:
```

Note

In order to use secure LDAP, the Java default truststore must be updated with the server's self signed certificate or the CA certificate for validating the server connection. The truststore should be updated before running the tool.

13.3.1.3. Discovery of UserSync Properties

Usersync-related properties are divided into two categories: User search related properties and group search related properties. This tool provides a `-d` option to discover user related and group related properties separately or all at once. The discover properties option is used as follows:

```
./run.sh -d <arg>
```

where `<arg>` can be

- `all` – discover all of the properties at once or
- `users` – discover only user search related properties or
- `groups` – discover only group search related properties

These properties are discovered based on the values provided in the input file for all of the mandatory properties.

The following are the user search related properties that are discovered using this tool:

1. Basic properties:
   - ranger.usersync.ldap.user.objectclass
   - ranger.usersync.ldap.user.groupnameattribute

- ranger.usersync.ldap.user.nameattribute

2. Advanced properties:

    - ranger.usersync.ldap.user.searchbase
    - ranger.usersync.ldap.user.searchfilter

Group search related properties that are discovered by this tool are as follows:

1. Basic properties:

    - ranger.usersync.group.searchenabled
    - ranger.usersync.group.objectclass
    - ranger.usersync.group.memberattributename
    - ranger.usersync.group.nameattribute

2. Advanced properties:

    - ranger.usersync.group.searchbase
    - ranger.usersync.group.searchfilter

Once all of the properties are discovered, the tool also retrieves the total count and details of first 20 users and/or groups and displays them in the output.

1. The value for the user search base is derived as the OU with max. no of users (from the first 20 users that are retrieved).
2. The value for the user search filter is derived as <user name attribute>=*
3. The value for the group search base is derived as the OU with max. no of groups (from the first 20 retrieved groups).
4. The value for the group search filter is derived as <group name attribute>=*

13.3.1.4. Discovery of Authentication Properties

The LDAP Connection Check tool provides a `-noauth` option to skip discovery of authentication properties. When this option is used, the tool will not suggest the values for authentication related properties.

```
./run.sh -noauth
```

If the LDAP server is of type active directory, the following properties are suggested:

- ranger.authentication.method
- ranger.ldap.ad.domain

If the LDAP server is not an active directory, the following properties are suggested:

- ranger.authentication.method

- ranger.ldap.user.dnpattern
- ranger.ldap.group.roleattribute
- ranger.ldap.group.searchbase
- ranger.ldap.group.searchfilter

These authentication properties can be discovered either by providing the values in the input file for only mandatory properties, or for all of the user and/or group related properties. After discovering the authentication properties, the tool also validates those properties by authenticating the given user, and reports authentication success or failure in the output.

13.3.1.5. Retrieval of Users and Groups

Usersync-related properties are divided into two categories: User search related properties and group search related properties. This tool provides a `-d` option to discover user related and group related properties separately or all at once. The discover properties option is used as follows:

```
./run.sh -r <arg>
```

where `<arg>` can be

- `users` – retrieve the total count and details of the first 20 users and associated groups, given the user search related properties in the input file.
- `groups` – retrieve the total count and details of the first 20 groups and associated users, given the group search related properties in the input file.
- `all` – retrieve both users and groups, given all of the corresponding properties in the input file.

13.3.1.6. Output Directory Content

This tool generates three files in the output directory specified with the `-o` option, or by default to the `/usr/hdp/current/ranger-usersync/ldaptool/output` directory.

- ambari.properties
- install.properties
- ldapConfigCheck.log

All of the discovered properties (related to usersync and/or authentication) are written to both the ambari.properties and install.properties files with the corresponding property names.

All of the other information, such as any retrieved users/groups, total count, authentication result, etc. are written to the ldapConfigCheck.log file. This log file also contains any errors or warnings generated while running the tool.

13.3.1.7. Other UserSync Related Properties

Some of the other usersync-related properties that are used by the tool and left with default values are:

- ranger.usersync.ldap.authentication.mechanism - Default authentication mechanism used is “simple”.
- ranger.usersync.pagedresultsenabled - Default is set to “true”.
- ranger.usersync.pagedresultssize - Default value for this property is “500”. This value can be tweaked depending on the bandwidth and resource availability in the deployment.
- ranger.usersync.ldap.username.caseconversion - Default value is set to “lower”
- ranger.usersync.ldap.groupname.caseconversion - Default value is set to “lower”
- ranger.usersync.ldap.user.searchscope - Default is set to “sub”. This value can be set to either “base” or “one” depending on how the user search is to be performed.
- ranger.usersync.group.searchscope - Default is set to “sub”. This value can be set to either “base” or “one” depending on how the group search is to be performed.

The following are the remaining usersync-related properties. These properties are not currently used by the tool and the values are left empty in the input file.

- ranger.usersync.credstore.filename - this property is unused as the tool supports only cleartext password.
- ranger.usersync.ldap.bindalias - this property is also not used by the tool.
- ranger.usersync.ldap.searchBase - This property is used as the user search base or group search base when they are not configured. Hence this value is left blank and not used by the tool.
- ranger.usersync.group.usermapsyncenabled - Mainly used for computing group memberships while retrieving users. Currently this value is set to “true”, but is not used by the tool.

13.3.1.8. Assumptions

Some properties are assumed to have one or more values as follows:

- User name attribute – **"sAMAccountName"** , **"uid"** , **"cn"**
- User Object class value – **"person"** , **"posixAccount"**
- User group member attribute – **"memberOf"** , **"ismemberOf"**
- Group Object class – **"group"** , **"groupOfNames"** , **"posixGroup"**
- Group name attribute – **"distinguishedName"** , **"cn"**
- Group member attribute – **"member"** , **"memberUid"**

13.3.1.9. Sample input.properties File

```
# Mandatory ldap configuration properties.
ranger.usersync.ldap.url=
ranger.usersync.ldap.binddn=
ranger.usersync.ldap.ldapbindpassword=

# Mandatory only for openLdap
```

```
ranger.usersync.ldap.user.searchbase=
ranger.usersync.ldap.user.searchfilter=

# For verifying authentication please provide sample username and password
ranger.admin.auth.sampleuser=
ranger.admin.auth.samplepassword=

# Optional properties will be determined based on the above search
# User attributes
ranger.usersync.ldap.user.nameattribute=
ranger.usersync.ldap.user.objectclass=
ranger.usersync.ldap.user.groupnameattribute=

# Group attributes
ranger.usersync.group.searchenabled=false
ranger.usersync.group.memberattributename=
ranger.usersync.group.nameattribute=
ranger.usersync.group.objectclass=
ranger.usersync.group.searchbase=
ranger.usersync.group.searchfilter=

# Other UserSync related attributes
ranger.usersync.ldap.authentication.mechanism=simple
ranger.usersync.pagedresultsenabled=true
ranger.usersync.pagedresultssize=500
ranger.usersync.ldap.username.caseconversion=lower
ranger.usersync.ldap.groupname.caseconversion=lower
ranger.usersync.ldap.user.searchscope=sub
ranger.usersync.group.searchscope=sub

ranger.usersync.credstore.filename=
ranger.usersync.ldap.bindalias=
ranger.usersync.ldap.searchBase=
ranger.usersync.group.usermapsyncenabled=false

# Authentication properties
ranger.authentication.method=
ranger.ldap.ad.domain=
ranger.ldap.user.dnpattern=
ranger.ldap.group.roleattribute=
ranger.ldap.group.searchbase=
ranger.ldap.group.searchfilter=
```

13.3.2. Install UserSync and Start the Service

Important

To ensure that LDAP/AD group level authorization is enforced in Hadoop, you should set up Hadoop group mapping for LDAP/AD.

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
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 encrypted 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)

Configuration Property Name	Default/Example Value	Required?
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)
SYNC_LDAP_SEARCH_BASE	Default is False. dc=example,de=com	N

SYNC_GROUP_SEARCH_ENABLED	Default is False.
	If set to True, and **SYNC_GROUP_USER_MAP_SYNC_ENABLED** is also set to True, you must set the following properties:
	`SYNC_GROUP_SEARCH_BASE=ou=`
	`People,dc=example,dc=com`
	`SYNC_GROUP_SEARCH_SCOPE=sub`
	`SYNC_GROUP_OBJECT_CLASS=`
	`groupofnames`
	`SYNC_LDAP_GROUP_SEARCH_FILTER=`
	`SYNC_GROUP_NAME_ATTRIBUTE=cn`
	`SYNC_GROUP_MEMBER_ATTRIBUTE_NAME=`
	`member`
	`SYNC_PAGED_RESULTS_ENABLED=true`
	`SYNC_PAGED_RESULTS_SIZE=`
	`500RANGER_`
	`SYNC_LDAP_REFERRAL=follow,`
	`ignore`
		N

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)

Configuration Property Name	Default/Example Value	Required?
logdir Location of the log directory where 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/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>
  ```

- Set up the JAVA_HOME environment variable to point to Java distribution on the installation machine.

  ```
  export JAVA_HOME=location-of-java-home-on-the-machine
  ```

- Edit the `install.properties` file in the `ranger-tagsync-install-directory` to support the operational environment.

- Keeping in mind the following two guidelines, edit the `install.properties` file in the `ranger-tagsync-install-directory` to add Audit to solr properties:

  - You must configure the XAAUDIT.SOLR.URL property based on your Solr installation. See `http://<solr_host>:8886/solr/ranger_audits` for details.

  - You must configure the XAAUDIT.SOLR.ZOOKEEPER property to NONE, if you are using stand alone Solr. or <zk1>:2181,<zk2>:2181/ranger_audits, using the correct zookeeper URL, if you are using SolrCloud.

  ```
  XAAUDIT.SOLR.ENABLE=true
  XAAUDIT.SOLR.URL=http://<solr_host>:8886/solr/ranger_audits
  XAAUDIT.SOLR.USER=NONE
  XAAUDIT.SOLR.PASSWORD=NONE
  XAAUDIT.SOLR.ZOOKEEPER=NONE
  XAAUDIT.SOLR.FILE_SPOOL_DIR=/var/log/hadoop/hdfs/audit/solr/spool
  ```

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 resource-based service and install the HDFS plug-in.

Install the HDFS Plug-in

1. Create an HDFS resource-based service in the Ranger Policy Manager. To do this, complete the HDFS Create Service screen, as described in the Configure an HDFS Service section of the *Hadoop Security Guide*.

   Make a note of the name you gave to this resource-based service; 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
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 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	60	Y

Configuration Property Name	Default/Example Value	Required?
Local audit log file writes are flushed to filesystem at regular flush interval		
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
XAAUIDT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ ranger_audits	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

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

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

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

4. To restart the service, issue the following commands:

```

su hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh stop
 namenode"
```

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

5. 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_5_0_0_1245-yarn-plugin-0.5.0.2.5.0.0-1245.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
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		

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
XAAUIDT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ ranger_audits	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	/etc/hadoop/conf/ranger-plugin-keystore.jks (default)	Only if SSL is enabled

Configuration Property Name	Default/Example Value	Required?
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_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 resource-based service name as in step 3.

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

10. **Optional**: If Wire Encryption is enabled after Ranger authorization is set up for YARN, follow these steps to update the resource manager URL in Ranger:

    a. Login to Ranger Admin as an admin user.

    b. Click 'Edit' for the YARN service. The service name is something like cl1_yarn (where 'cl1' is the name of the cluster).

    c. Update property 'YARN REST URL', with the https URL for the resource manager.

    d. Click 'Save.'

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_5_0_0_1245-kafka-plugin-0.5.0.2.5.0.0-1245.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/2.5.0.0-1245/kafka	Y
POLICY_MGR_URL URL for policy admin	http://<FQDN of ranger admin host>:6080	Y
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

Configuration Property Name	Default/Example Value	Required?
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
XAAUDIT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ranger_audits	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 events 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

Configuration Property Name	Default/Example Value	Required?
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 resource-based service 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/2.5.0.0-1245/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 resource-based service
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 resource-based service in the Ranger Policy Manager. To do this, complete the HBase Create Service screen, as described in the Configure an HBase Service section of the *Hadoop Security Guide*.

   Make a note of the name you gave to this resource-based service; 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		
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	FALSE (default)	Y

Configuration Property Name	Default/Example Value	Required?
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.		
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 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 HDSF audit log open() call fails, it will be re-tried at this interval	60	Y

Configuration Property Name	Default/Example Value	Required?
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.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ranger_audits	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		
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 resource-based service .

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 resource-based service. To create the Hive resource-based service, complete the Hive Create Service screen as described in the Configure a Hive Service section of the *Hadoop Security Guide*.

   Make a note of the name you gave to this resource-based service; 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		

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) 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
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

Configuration Property Name	Default/Example Value	Required?
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
XAAUDIT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ ranger_audits	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 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/hive/conf/ranger-plugin-truststore.jks* (default)	If SSL is enabled

Configuration Property Name	Default/Example Value	Required?
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 resource-based service.

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 resource-based service. To do this, complete the Knox Create Service screen as described in the Configure a Knox Service section of the *Hadoop Security 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 resource-based service; 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 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

Configuration Property Name	Default/Example Value	Mandatory?
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
XAAUDIT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ ranger_audits	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 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-keystore.jks*	If SSL is enabled

Configuration Property Name	Default/Example Value	Mandatory?
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:

```
su knox -c "/usr/hdp/current/knox-server/bin/gateway.sh stop"
su knox -c "/usr/hdp/current/knox-server/bin/gateway.sh start"
```

7. Optionally, add a new user by following these steps. This is required only if you want users to verify the communication between the Ranger Knox plug-in and Ranger `admin`.

   a. Add the following information in the `usertest` block in the `/usr/hdp/current/knox-server/templates/users.ldif` file, assuming that Knox is using the demo LDAP. If Knox is configured with a different LDAP, you need to add the information in the `users.ldif` file at that location.

      - # entry for sample user usertest
      - dn: uid=usertest,ou=people,dc=hadoop,dc=apache,dc=org
      - objectclass:top
      - objectclass:person
      - objectclass:organizationalPerson
      - objectclass:inetOrgPerson
      - cn: usertest
      - sn: usertest

- uid: usertest
- userPassword:usertest-password

b. Restart ldap. Enter the following command:

```
./usr/hdp/current/knox-server/bin/ldap.sh stop / start
```

c. Verify in which topology the Knox plugin is enabled and then run the command accordingly. Typically, the topology is `admin`.

d. Issue the **curl** command to check policy enforcement for Knox. Typically, the *`topology`* in the command is `admin`.

```
curl -iku admin:admin-password -X GET 'https://<knox_host>:8443/gateway/
<topology>/webhdfs/v1?op=LISTSTATUS'
```

e. Create a certificate for the test connection to be successful. Follow these steps to import a Knox SSL certificate in truststore used by Ranger admin:

i. Login to the machine running Knox.

ii. Export the Knox certificate:

```
cd $GATEWAY_HOME/data/security/keystores
keytool -exportcert -alias gateway-identity -keystore gateway.jks -file
knox.crt
```

Typically `$GATEWAY_HOME/data/security/keystores` is `/usr/hdp/current/knox-server/data/security/keystores` on a Linux machine.

iii. Copy knox.crt onto the machine running Ranger admin to a working directory, for example, `/etc/ranger/admin`.

iv. Replicate cacerts bundled with the JDK:

```
cd /etc/ranger/admin
cp <JDK_HOME>/jre/lib/security/cacerts cacertswithknox
```

v. Import the Knox certificate into the replicated new keystore.

```
keytool -import -trustcacerts -file <knox.crt
 created above>  -alias knox  -keystore cacertswithknox
password: changeit
```

vi. Edit /usr/hdp/current/ranger-admin/ews/ranger-admin-services.sh and add the parameter *`-Djavax.net.ssl.trustStore=<path to the cacertswithknox>`* to the *`JAVA_OPTS`* parameter in the script.

vii.Restart Ranger Admin.

8. 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 resource-based service.
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 resource-based service, as described in the Configure a Storm Service section of the *Hadoop Security Guide*.

   Make a note of the name you gave to this resource-based service; 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	false (default) true	Y

Configuration Property Name	Default/Example Value	Mandatory?
logs from the UI, set this value to TRUE.		
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE, PostgreSQL, 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 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.	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

Configuration Property Name	Default/Example Value	Mandatory?
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
XAAUDIT.SOLR.ENABLE	true	Y
XAAUDIT.SOLR.URL	http://<solr_host>:8886/solr/ranger_audits	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/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. Installing and Configuring Tag-Based Access Control

Ranger 0.6 supports authorization of access based on tags associated with resources, in addition to resource-based access authorization. The tag-based policy model offers many advantages over the resource-based authorization model. One of the important advantages of this model is that it supports separation of resource-classification from access-authorization. This, in turn, allows security administrators to conceptualize and author access policies across multiple components (such as HDFS, Hive) in terms of type/class of data.

Ranger includes a module, called Tag Synchronization Module (Tag-Sync) that enables synchronization of tagging information (such as entity-tag association as well value of tag-attributes if any) between a tag-source (Atlas) and Ranger Admin. To use tag-based access, you must perform the following:

- Install the Tag Synchronization Module
- Configure Kafka Notifications
- Validate the Installation

13.5.1. Installing the Tag Synchronization Module

To install the tag synchronization module:

1. Log in to the host machine as super-user (root).

2. Locate and download the ranger-tagsync package, then install it by executing `yum install ranger-tagsync`.

   Installing the ranger-tagsync package will unpack and create a directory (`ranger-tagsync-install-directory`) and needed sub-directories and files in the `ranger-tagsync` directory under the Apache installation root directory.

3. Make the `ranger-tagsync-install-directory` your current working directory.

   ```
   cd ranger-tagsync-install-directory
   ```

4. Set up the JAVA_HOME environment variable to point to Java distribution on the installation machine.

   ```
   export JAVA_HOME=location-of-java-home-on-the-installation-
   machine
   ```

5. Edit the `install.properties` file in the `ranger-tagsync-install-directory` to reflect the operational environment.

Table 13.10. install.properties File Properties

Property	Description	Comments
TAGADMIN_ENDPOINT	URL of the destination of the tags	Customize the value to reflect your deployment
TAGADMIN_SSL_CONFIG_FILENAME	File containing SSL configuration	Customize the value to point to SSL configuration specific to your deployment. The value will be ignored in the scheme in TAGADMIN_ENDPOINT URL is "http".
TAGSYNC_KEYSTORE_FILENAME	File to store encrypted password for Ranger Admin user 'rangertagsync' that is used for communicating with TagAdmin	Customize the value to suit your deployment. If there is no value specified then the password will not be encrypted.
TAG_SOURCE	Source of the tagging information	At this time, the only supported value is 'Atlas'.
TAGSYNC_ATLAS_KAFKA_ENDPOINTS	URL of the Kafka endpoint to which Atlas sends its notifications	Customize the value to reflect your deployment.
TAGSYNC_ATLAS_ZOOKEEPER_ENDPOINT	URL of the Zookeeper endpoint needed for Atlas	Customize the value to reflect your deployment.
TAGSYNC_ATLAS_CONSUMER_GROUP	String representing Kafka Consumer Group ID used by Tag-Sync	Customize the value to reflect your deployment.
TAGSYNC_ATLAS_TO_RANGER_SERVICE_MAPPING	Mapping between Atlas cluster-name, component-type, and Ranger service-name	This mapping is provided as a string value in the following format. *clusterName,componentType,serviceName;clusterName2,co* Customize the value to reflect your deployment.
unix_user	ranger-tagsync process is run as this Unix user.	There is no need to change this value (default is 'ranger').
unix_group	ranger-tagsync process is run with this Unix group-id.	There is no need to change this value (default is 'ranger').
logdir	Directory where logs are store	Customize the value to suit your deployment.

6. Run the setup script to create the run-time directory structure.

```
./setup.sh
```

The run-time directory structure copies scripts into appropriate directories, creates files required for Atlas integration, creates configuration files which are used by the ranger-tagsync process to configure itself, and sets up symbolic links wherever necessary.

Table 13.11. Run-time Directory Structure

Directory	File	Linked to
/etc/ranger/tagsync/conf	*ranger-tagsync-install-directory*/conf	
/etc/ranger/tagsync/conf	application.properties	Properties required for Atlas interface
/etc/ranger/tagsync/conf	java_home.sh	Shell command to set JAVA_HOME environment variable
/etc/ranger/tagsync/conf	log4j.xml	Log4j configuration properties

Directory	File	Linked to
/etc/ranger/tagsync/conf	ranger-tagsync-site.xml	Ranger-tagsync process configuration properties
/etc/ranger/tagsync/conf	rangertagsync.jceks	Encrypted password for Ranger Admin user ‘rangertagsync’
/etc/init.d	ranger-tagsync	Shell script to start/stop/query ranger-tagsync service
/usr/bin	ranger-tagsync-services.sh	*ranger-tagsync-install-directory*/ ranger-tagsync.sh
/var/log/ranger/tagsync	*ranger-tagsync-install-directory*/log	Directory where logs (ranger-tagsync.log*) generated by ranger-tagsync are stored.
/var/run/ranger	tagsync.pid	File containing process-id of the ranger-tagsync process, if it is running

13.5.2. Configuring Kafka Notifications

Ranger receives notifications of entity changes from Atlas via Kafka. To generate these notifications, you must configure Atlas properties that pertain to ranger-tagsync. For instructions on how to configure the Kafka notifications, see Configuring Atlas for Kafka Notifications .

13.5.3. Validating the Tag-Sync Module Installation

Important

Hortonworks recommends ensuring that all plug-ins are configured and their corresponding services (aka repositories) are provisioned in Ranger-Admin before starting the Tag-Sync module. Starting the Tag-Sync module before configuring all plug-ins could result in some of the tag-related notifications not being captured by Ranger-Admin.

To check if ranger-tagsync is running, log in as super-user and enter the following:

```
service ranger-tagsync status
```

To start the ranger-tagsync process, log in as super-user and enter the following:

```
service ranger-tagsync start
```

To stop the ranger-tagsync process, log in as super-user and enter the following:

```
service ranger-tagsync stop
```

13.5.4. Optional: Updating Password for Ranger Admin User rangertagsync

If, after installation, you want to change the password of ‘rangertagsync’ user, complete the following steps:

1. Change the password using Ranger Admin.

2. If the password is stored in encrypted form, run the `updatetagadminpassword.sh` script in the `ranger-tagsync-install-directory` to update the password in the TAGSYNC_KEYSTORE_FILENAME keystore file.

   Alternatively, you can modify the `/etc/ranger/tagsync/conf/ranger-tagsync-site.xml` file manually to update the `ranger.tagsync.tagadmin.password` property with your updated password.

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
- 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

Hue is a Web UI for Hadoop.

Hue supports the following features:

- Beeswax to execute Hive queries
- FileBrowser to access HDFS
- HCatalog application for Hive metadata and table management
- Pig application to execute Pig queries
- Job Designer to create MapReduce/Streaming/Java jobs
- Oozie application to submit and schedule workflows
- JobBrowser for view MapReduce jobs

This chapter describes the basics of installing, configuring, and validating Hue.

1. Before You Begin
2. Configure HDP to Support Hue
3. Install the Hue Package
4. Configure Hue to Communicate with the Hadoop Components
5. Configure Hue for Databases
6. Start, Stop, and Restart Hue
7. Validate the Hue Installation

14.1. Before You Begin

Before you begin your Hadoop installation, you must meet the following requirements.

Note: Hue can only connect to one Hadoop cluster.

1. Ensure that you are using an operating system that supports Hue.

   The following 64-bit operating systems support Hue:

   - CentOS 6
   - Oracle Linux 6
   - Red Hat Enterprise Linux (RHEL) 6
   - SUSE Linux Enterprise Server SLES

2. Ensure that you are using a browser that can access Hue.

   While Hue only runs on Linux, you can access Hue using browsers that run on Linux, Windows, or Mac.

   See Browser Requirements in the *Automated Install with Ambari Guide* for a complete list of browsers that work with Hue.

3. Ensure that you are using database that supports Hue.

   Refer to Supported Database Matrix for the Hortonworks Data Platform for a complete list of supported databases.

4. Ensure that you have Python 2.6.6 or higher installed.

5. Ensure that you have at least core Hadoop on your system. See Configure the Remote Repositories for more information.

6. Ensure that 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`, yum cannot locate a matching RPM. This can happen if the repository hosting the HDP RPMs is unavailable, or has been disabled. If this occurs, follow the instructions at Configure the Remote Repositories to configure either a public or private repository before proceeding.

14.2. Configure HDP to Support Hue

For Hue to communicate properly with HDP components, some minor configuration changes of your HDP cluster are required.

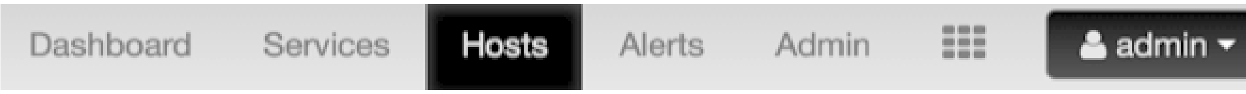
Complete the instructions in this section using Ambari. Do not edit the configuration files directly.

Use Ambari to start and stop the services.

1. Use the `admin` account, login to the Ambari Web UI at **http://localhost.com:8080** or or **http://127.0.0.1:8080**.

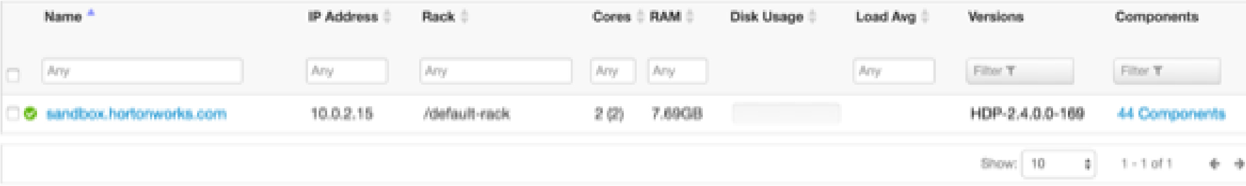
2. Stop the namenode.

   a. Select `Hosts` on the Navigation Header.



b. Select the FQDN.

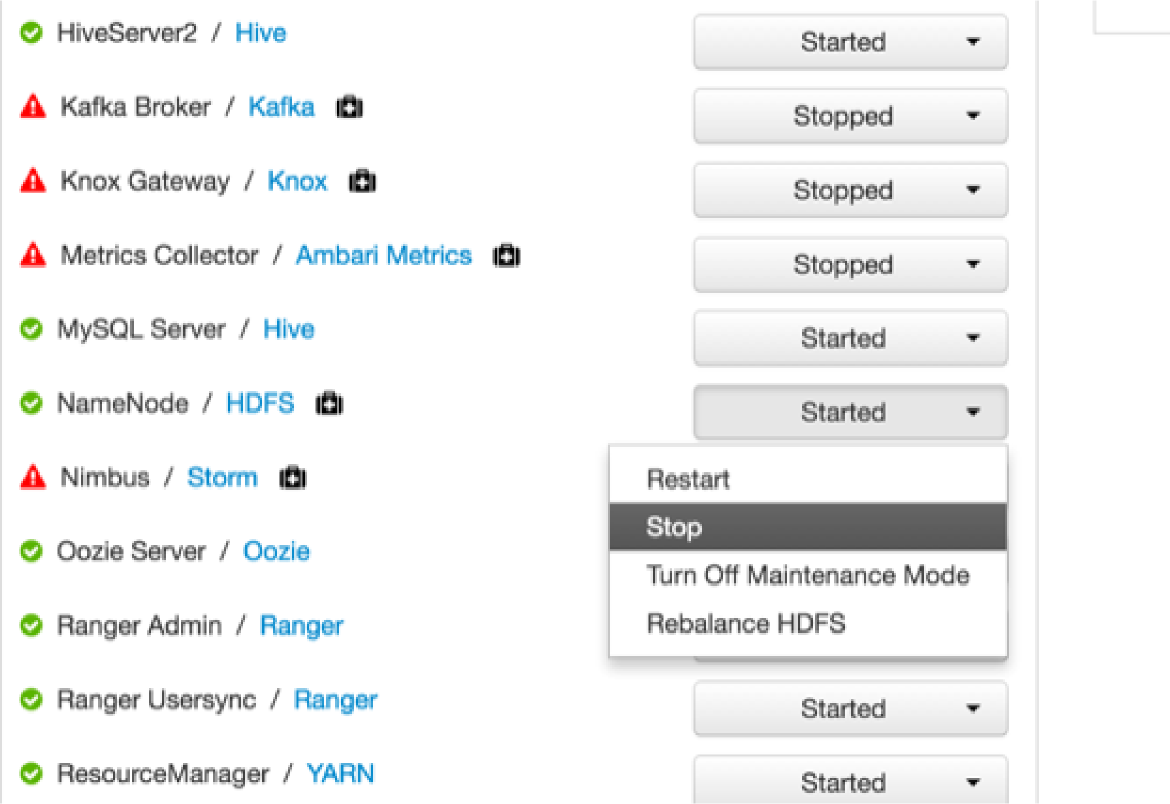
In this example, `sandbox.hortonworks.com` is the FQDN.



	Name	IP Address	Rack	Cores	RAM	Disk Usage	Load Avg	Versions	Components
□	Any	Any	Any	Any	Any		Any	Filter	Filter
□ ✓	sandbox.hortonworks.com	10.0.2.15	/default-rack	2 (2)	7.69GB			HDP-2.4.0.0-169	44 Components

Show: 10 1 - 1 of 1

c. Scroll down to `NameNode` and click on `Started` to reveal the drop down menu. Select `Stop` from the drop down menu to stop the Namenode.

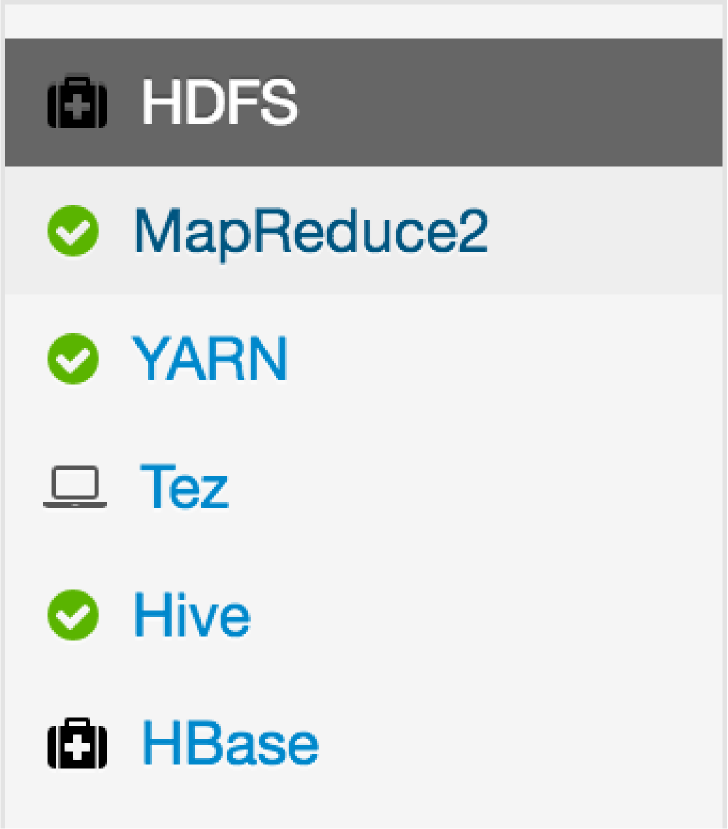


✓ HiveServer2 / Hive	Started
⚠ Kafka Broker / Kafka	Stopped
⚠ Knox Gateway / Knox	Stopped
⚠ Metrics Collector / Ambari Metrics	Stopped
✓ MySQL Server / Hive	Started
✓ NameNode / HDFS	Started
⚠ Nimbus / Storm	
✓ Oozie Server / Oozie	
✓ Ranger Admin / Ranger	
✓ Ranger Usersync / Ranger	Started
✓ ResourceManager / YARN	Started

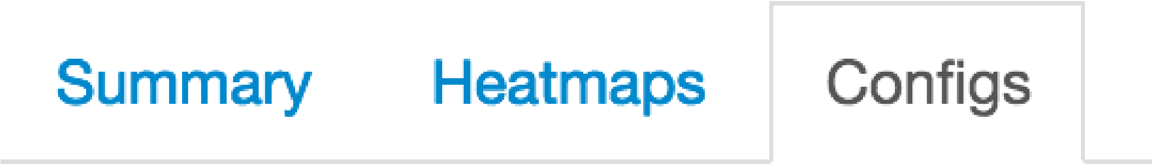
d. Click `OK` to confirm.

3. Modify `hdfs-site` settings.

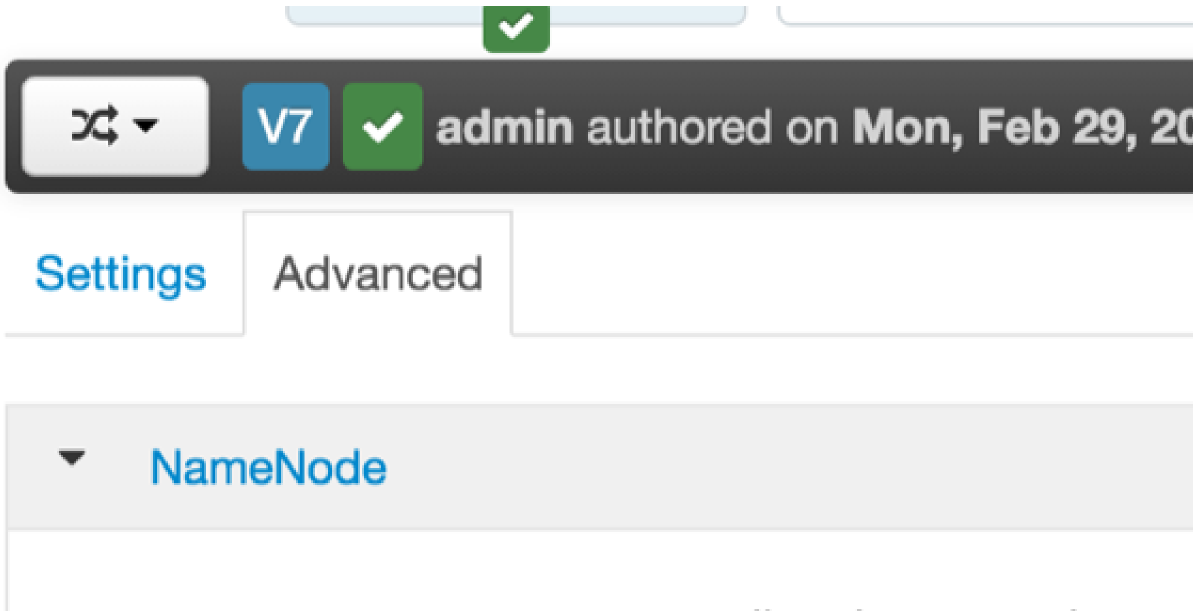
a. Click on `HDFS` from the `Services` menu on the left side of the screen.



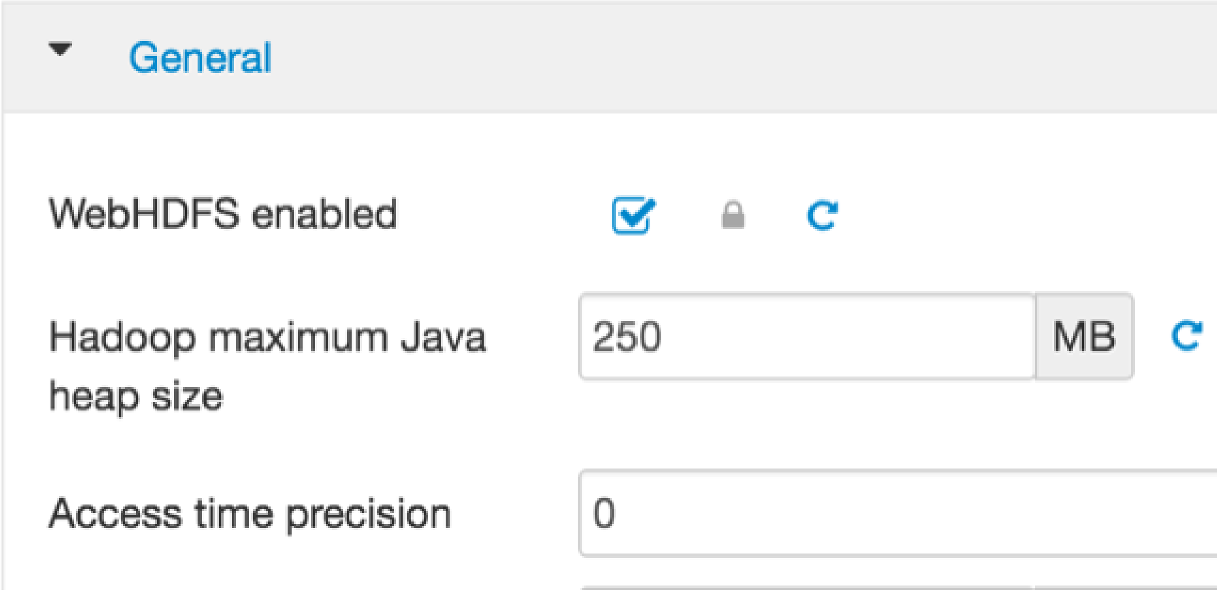
b. Click the `Configs` tab.



Click `Advanced`.



c. Scroll down to the `General` settings. Ensure that the `WebHDFS enabled` checkbox is checked.



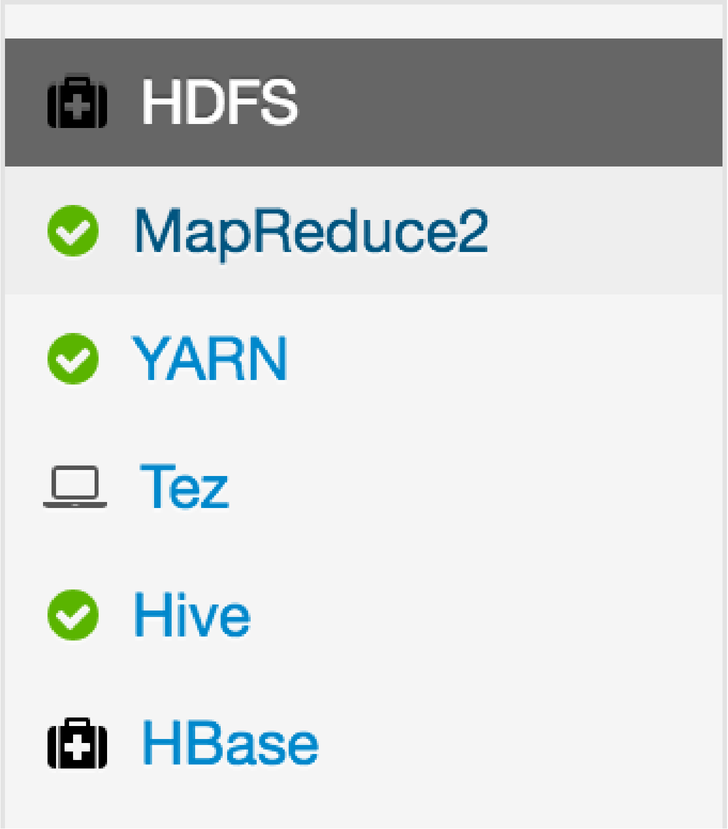
WebHDFS enabled ☑

Hadoop maximum Java heap size 250 MB

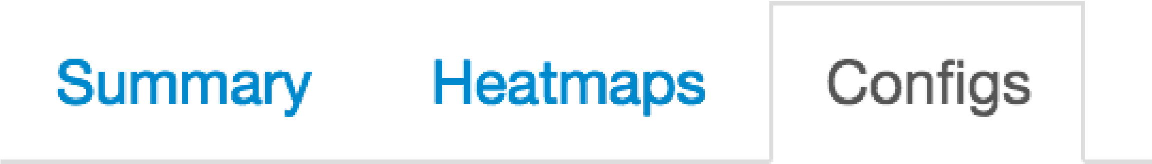
Access time precision 0

4. Modify the `core-site` settings.

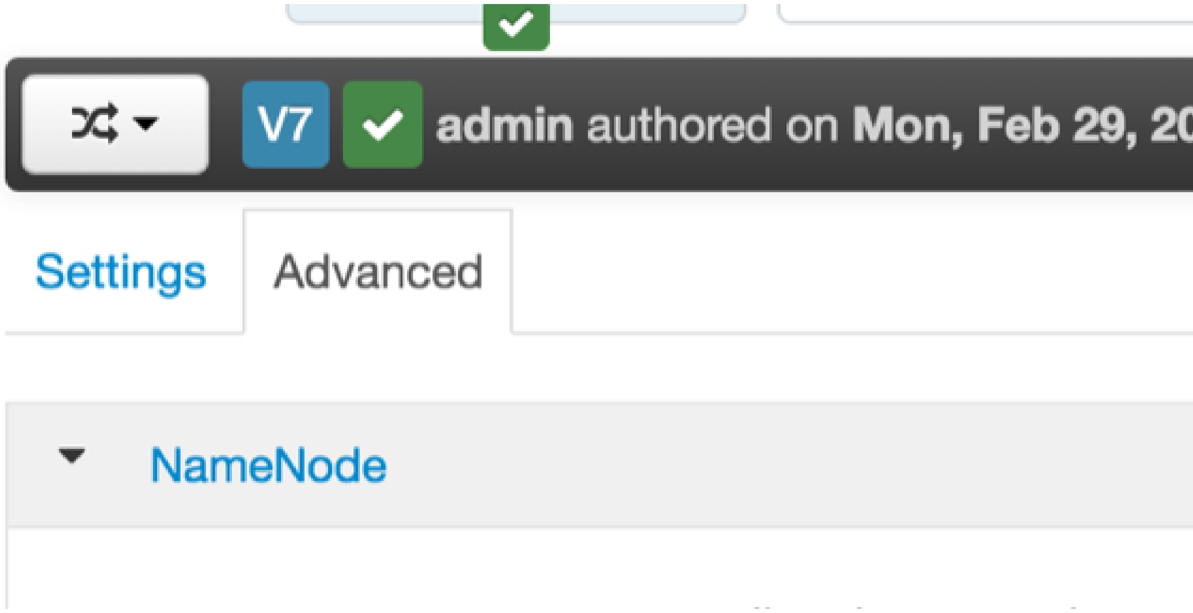
   a. Click on HDFS from the `Services` menu on the left side of the screen.



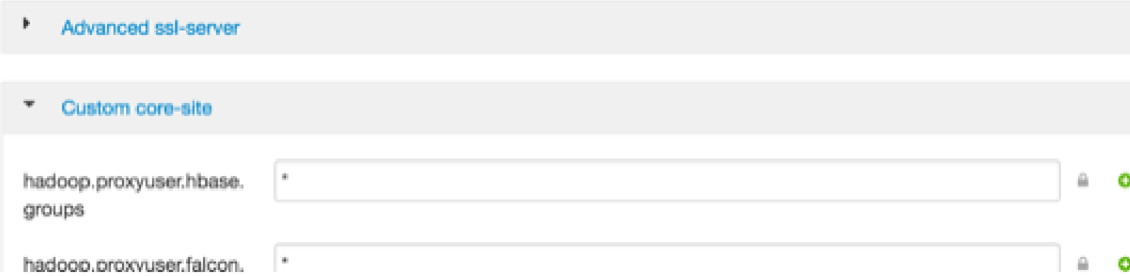
b. Click the `Configs` tab.



Click `Advanced`.



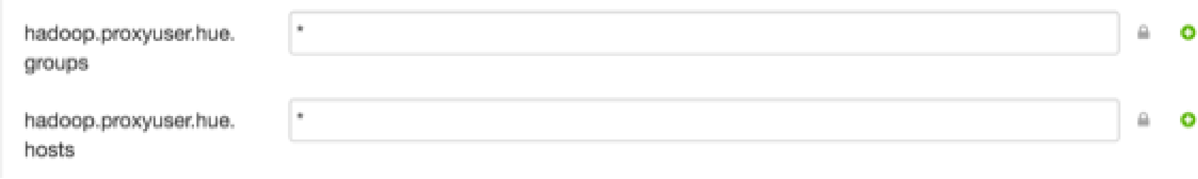
c. Scroll down to `Custom core-site` settings.



hadoop.proxyuser.hbase.groups *

hadoop.proxyuser.falcon. *

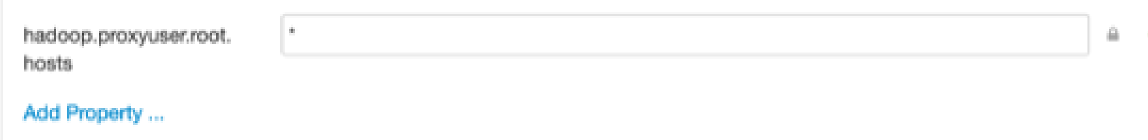
d. Ensure the `hadoop.proxyuser.hue.groups` and `hadoop.proxyuser.hue.hosts` properties and their respective values are set.



hadoop.proxyuser.hue.groups *

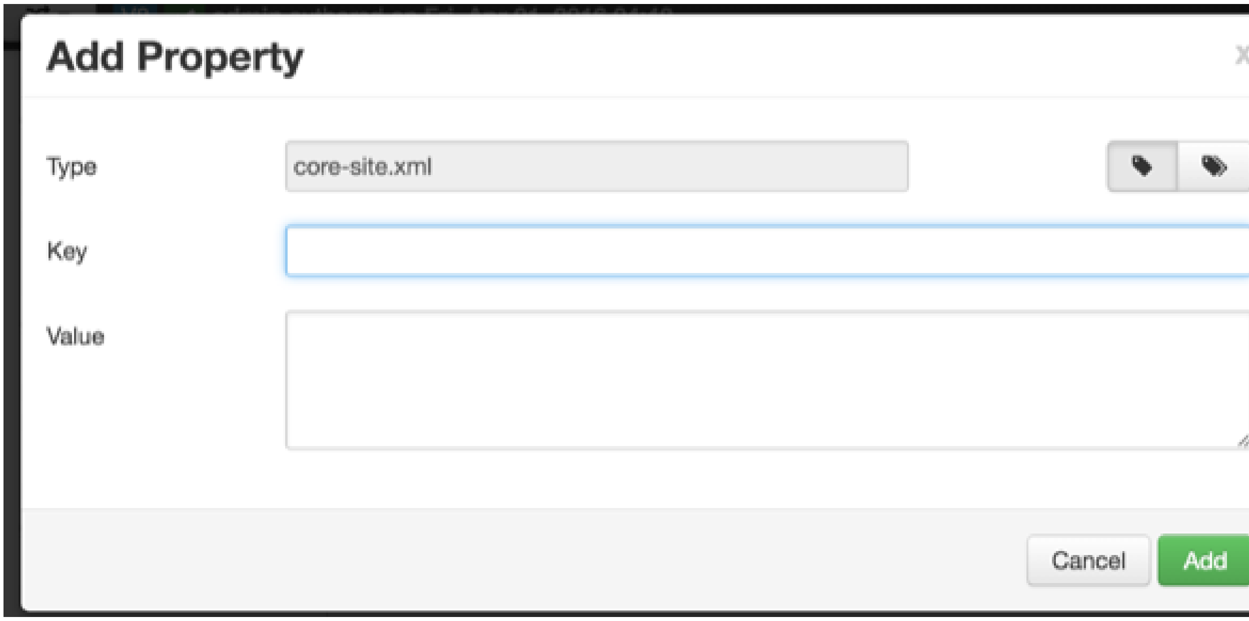
hadoop.proxyuser.hue.hosts *

e. If they are not, add them, by clicking on `Add Property` ....



hadoop.proxyuser.root.hosts *

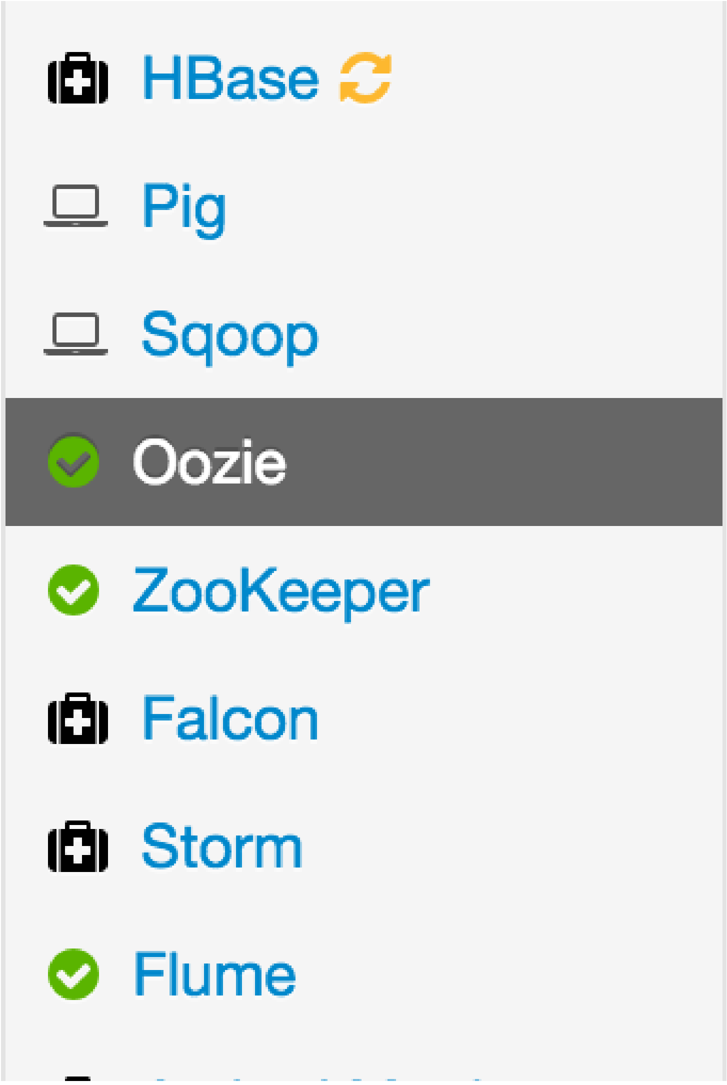
f. Set the Key as the setting name (for example, `hadoop.proxyuser.hue.hosts`) and the `Value` as *value*, for example, `*`. Click Add.



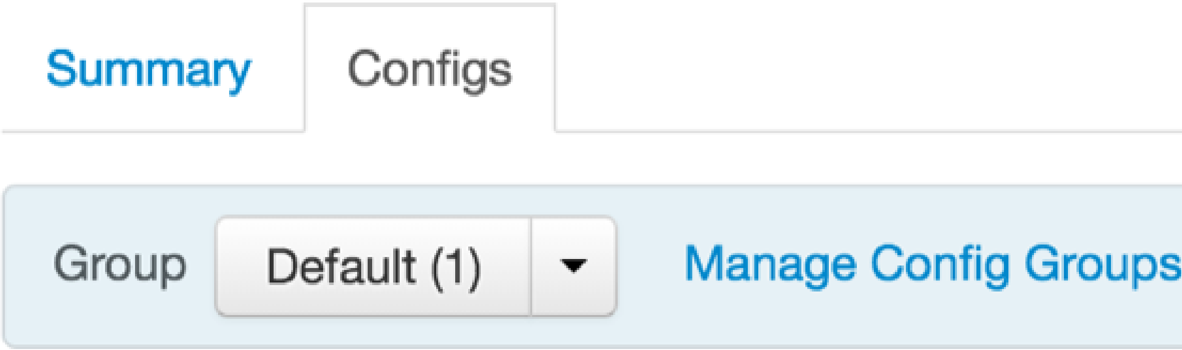
core-site.xml

5. Modify `oozie-site` settings

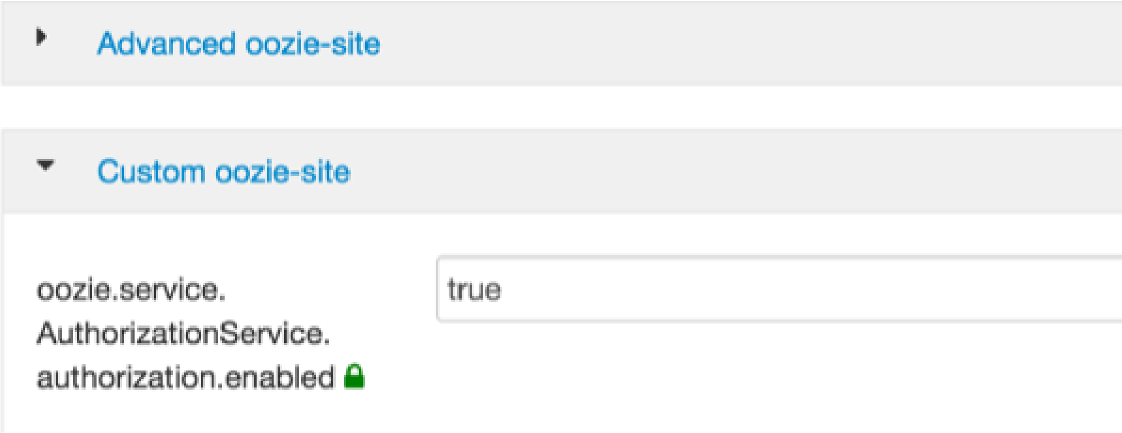
   a. From the `Services` menu, click `Oozie`.



b. Click `Configs`.

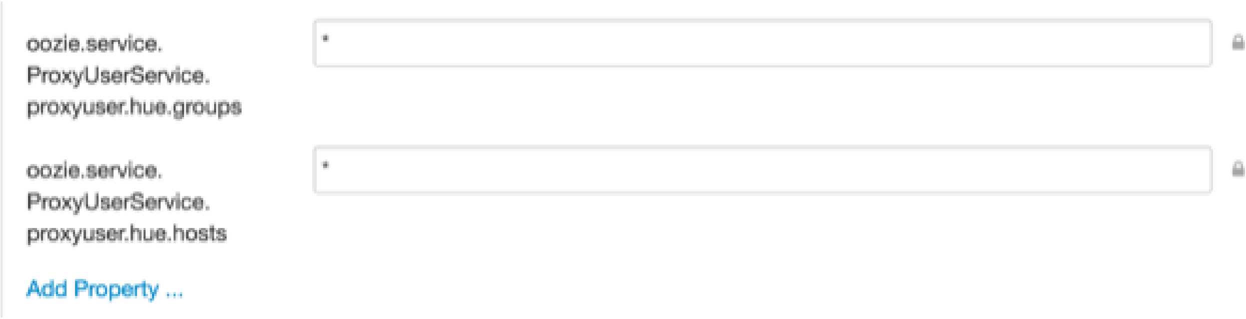


c. Scroll down to `Custom oozie-site`.



oozie.service.AuthorizationService.authorization.enabled: true

d. Ensure that the `oozie.service.ProxyUserService.proxyuser.hue.groups` and the `oozie.service.ProxyUserService.proxyuser.hue.hosts` properties are present.



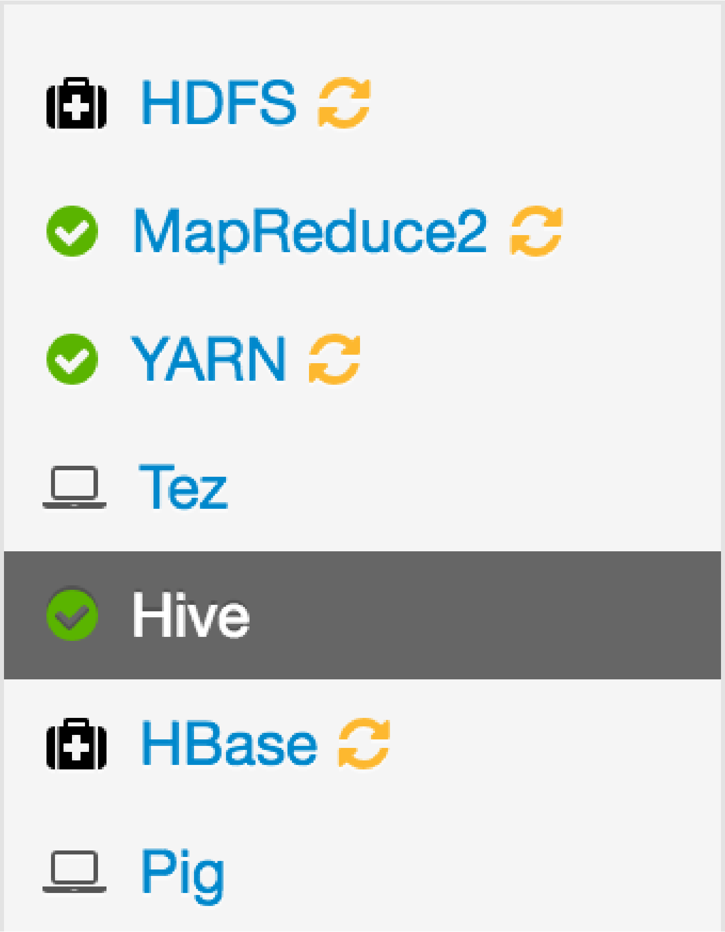
oozie.service.ProxyUserService.proxyuser.hue.groups: *

oozie.service.ProxyUserService.proxyuser.hue.hosts: *

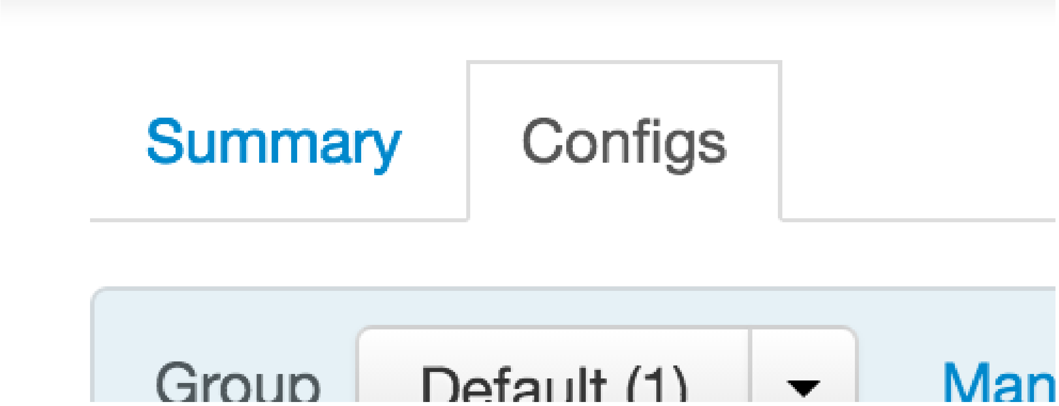
e. If they are not present, add them by clicking on `Add Property` ....

6. Modify `hive-site` settings.

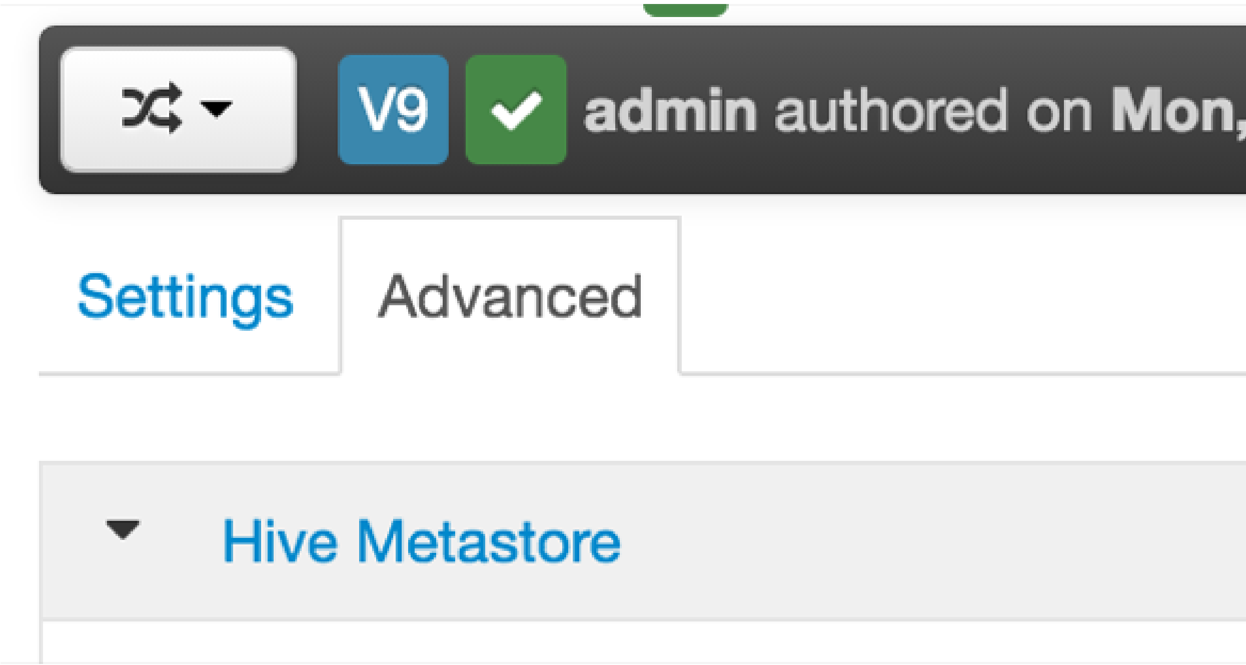
   a. From the `Services` menu click on `Hive`.



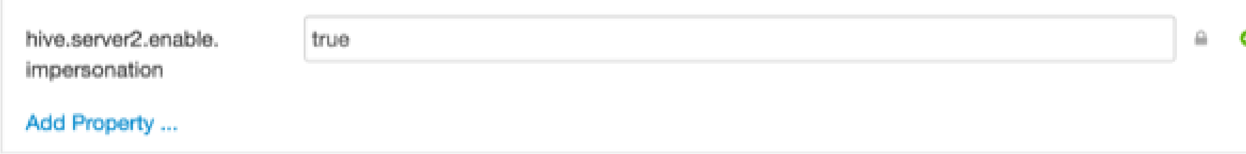
b. Click the `Configs` tab.



c. Click Advanced.



d. Ensure the `hive.server2.enable.impersonation property` is present.



hive.server2.enable.impersonation: true

e. If the `hive.server2.enable.impersonation property` is not present, click on `Add Property` ... and add it.

14.3. Install the Hue Packages

Hue server installation assumes the installation of the following sets of packages: hue, `hue-common`, `hue-server`, `hue-beeswax`, `hue-oozie`, `hue-pig`, and .

By default, the hue meta-package installs the package and all Hue applications as its dependencies.

Follow these steps to install the Hue packages:

1. Stop all of the services in your cluster.

   For more information, see Stopping HDP Services in the *HDP Reference Guide*.

2. Choose a Hue Server host machine in your cluster on which to deploy your Hue server.

   - You can deploy Hue on any host in 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.

3. Configure the firewall to ensure that all ports are open.

   - 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.

4. Install Hue.

- If you are running CentOS 6, RHEL 6, or Oracle Linux 6, run this command on all Hue Server host machines:

```
yum install hue
```

- If you are running SLES, run this command on all Hue server host machines:

```
zypper install hue
```

14.4. Configure Hue to Communicate with the Hadoop Components

14.4.1. Configure the Web Server

These configuration variables are in the `[desktop]` section of the `/etc/hue/conf/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 in the example 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.

   a. Configure Hue to use your private key.

      Add the following to the /etc/hue/conf/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

To upload files using the Hue File Browser over HTTPS, you must have a proper SSL Certificate. 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 -shal -key host.key > host.cert
```

14.4.2. Configure Hadoop

These configuration variables are in the `[hadoop]` section of the `/etc/hue/conf/hue.ini` configureation file:

1. Configure an HDFS cluster.

   Hue only supports one HDFS cluster. Ensure that you define the HDFS cluster under the [hadoop][[hdfs_clusters]] [[[default]]] subsection of the `/etc/hue/config/hue.ini` configuration file.

   Use the following variables to configure the 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.

Ensure that you define the YARN cluster under the `[hadoop][[yarn_clusters]] [[[default]]]` sub-section of the `/etc/hue/config/hue.ini` configuration file.

For more information regarding how to configure Hue with a NameNode HA cluster see see Deploy Hue with a ResourceManager HA Cluster in the *High Availabiltiy for Hadoop Guide*.

Use the following variables to configure a YARN cluster:

Variable	Description	Default/Example Value
submit_to	Set this property to true. Hue submits jobs to this YARN cluster. Note that JobBrowser is not 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

3. Configure Beeswax

In the `[beeswax]` section of the of the `/etc/hue/config/hue.ini` configuration file, you can specify the following values:

Variable	Description	Default/Example Value
hive_server_host	Host where Hive server Thrift daemon is running. If Kerberos security is 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 might fail with an internal error processing query message.

Look for an error message java.lang.OutOfMemoryError:

GC overhead limit exceeded in the beeswax_serer.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`.

4. Configure HiverServer2 over SSL (Optional)

Make the following changes to the /etc/hue/conf/hue.ini configuration file to configure Hue to communicate with HiverServer2 over SSL:

```
[[ssl]]
SSL communication enabled for this server.
enabled=falsePath to Certificate Authority certificates.
cacerts=/etc/hue/cacerts.pemPath to the public certificate file.
cert=/etc/hue/cert.pemChoose whether Hue should validate certificates
 received from the server.
validate=true
```

5. Configure JobDesigner and Oozie

   In the `[liboozie]` section of the `/etc/hue/conf/hue.ini` configuration file, specify the `oozie_url`, the URL of the Oozie service as specified by the OOZIE_URL environment variable for Oozie.

6. Configure WebHCat

   In the `[hcatalog]` section of the `/etc/hue/conf/hue.ini` configuration file, set `templeton_url`, to the hostname or IP of the WebHCat server. An example could be `http:// hostname:50111/templeton/v1/`.

14.5. Configure Hue for Databases

By default, Hue uses an embedded database, SQLite 3.6.20. If you are using SQLite 3.6.20, there is no need to configure Hue for databases.

Alternatively, you can configure Hue to use any of the following external databases:

- Using Hue with Oracle
- Using Hue with MySQL
- Using Hue with PostgreSQL

14.5.1. 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 the Oracle 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 a symbolic link for library expected by `cx_Oracle`:

   ```
   ln -sf libclntsh.so.11.1 libclntsh.so
   ```

   e. Install the `cx_Oracle python` module.

   f. Confirm that the python-setuptools are present on the Hue node, for example:

   ```
   rpm -qa | grep python-setuptools
   ```

   If the phython-setuptools are not present, install them, using the following command:

   ```
   yum install python-setuptools
   ```

   g. Install the cx_Oracle module:

   ```
   /usr/lib/hue/build/env/bin/pip install cx_Oracle
   ```

   h. 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:

```

cd /usr/lib/hue
source build/env/bin/activate
hue syncdb --noinput
hue migrate
deactivate
```

5. Populate /usr/lib64 with Oracle instant-client library files.

6. Copy the *.so.* files from oracle instantclient directory path to /usr/lib64.

14.5.2. 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;
```

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:

```
[[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.

```
cd /usr/lib/hue
source build/env/bin/activate
hue syncdb --noinput
hue migrate
deactivate
```

14.5.3. 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. 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:

   ```
   cd /usr/lib/hue
   source build/env/bin/activate
   hue syncdb --noinput
   hue migrate
   deactivate
   ```

14.6. Start, Stop, and Restart Hue

Use the following commands to start, stop, and restart hue.

1. To start Hue, run the following command on the Hue server:

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

2. To stop Hue, run the following command on the Hue server:

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

3. If you make changes to /etc/hue/conf/hue.ini you need to restart Hue.

   To restart Hue, run the following command on the Hue server:

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

14.7. Validate the Hue Installation

Ensure that Hue is properly installed.

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

1. Also to make sure that Hue Server was configured properly, select **About > Check** for misconfiguration or **http://hue.server:8000/debug/check_config**. If there was any potential misconfiguration detected, then you need to fix this and to restart Hue.

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:

- Install the Sqoop Package
- Set Up the Sqoop Configuration
- Validate the Installation

15.1. Install the Sqoop 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 sqoop
   ```

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

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

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

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

Installation

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

- For RHEL/CentOS/Oracle Linux:

  ```
  yum install sqoop
  ```

- For SLES:

  ```
  zypper install sqoop
  ```

- For Ubuntu:

  ```
  apt-get 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 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 is to 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.

- Install Mahout
- Validate Mahout

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
  ```

- For Ubuntu and Debian:

  ```
  apt-get 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:

- Understanding Flume
- Installing Flume
- Configuring Flume
- Starting Flume
- HDP and Flume
- A Simple Example

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.

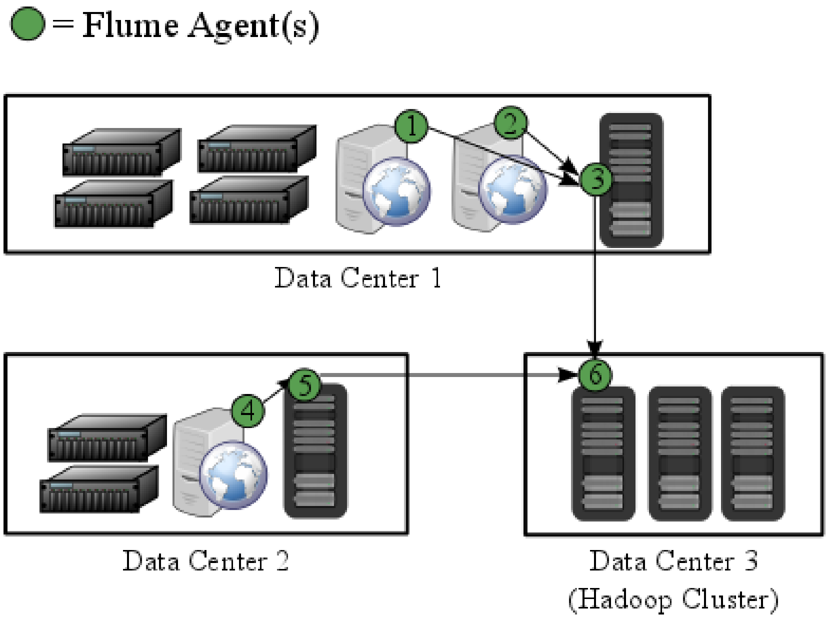
17.2. Installing Flume

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-2.5
   ```

   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 2.7.3 with HDP patches
HBase Sink	HBase 1.1.2 with HDP patches
Hive Sink	Hive 1.2.1000, HCatalog 1.2.1000, and Hadoop 2.7.3 (with HDP patches)

Installation

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

To install Flume from a terminal window, type:

- For RHEL or CentOS:

  ```
  yum install flume
  ```

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

- For SLES:

  ```
  zypper install flume
  ```

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

- For Ubuntu and Debian:

  ```
  apt-get install flume
  ```

  ```
  apt-get 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.

1. Install the Storm Package
2. Configure Storm
3. Configure a Process Controller
4. (Optional) Configure Kerberos Authentication for Storm
5. (Optional) Configuring Authorization for Storm
6. Validate the Installation

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
     ```

   - For Ubuntu and Debian:

     ```
     apt-get install storm
     ```

   ### Important

   Ubuntu and Debian users must manually create the `/etc/storm/conf` directory and the storm.yaml file that resides there.

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

In Storm 1.0 , Java package naming moved from backtype.storm to org.apache.storm.

If you intend to run any topologies that used to run previous versions of Storm in Storm 1.0, you can do so by using one of two options:

- You can rebuild the topology by renaming the imports of the backtype.storm package to org.apache in all of your topology code.

or

- You can add config `Client.jartransformer.class = org.apache.storm.hack.StormShadeTransformer` to `storm.yaml`.

Either of these configurations allows the storm jar to transform all of the `backtype.storm` imports in your topology to `org.apache.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 `/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:

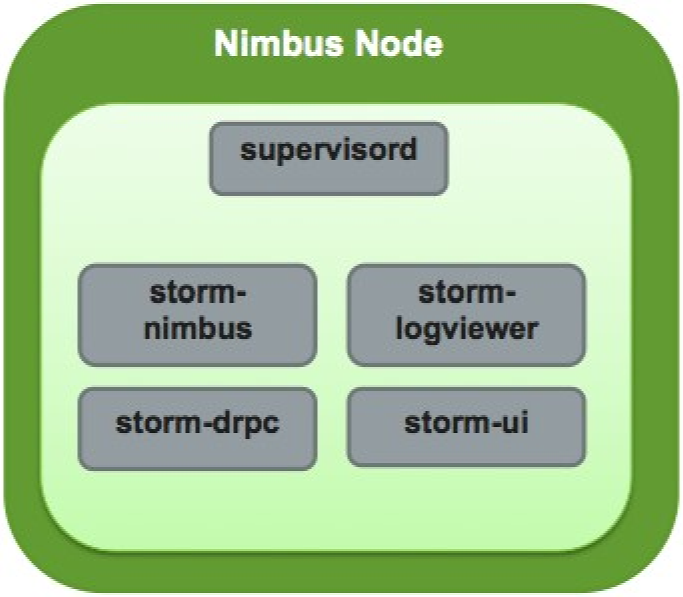
   ```
   chown -R storm:storm $STORM_LOCAL_DIR
   ```

```
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:
"org.apache.storm.security.auth.kerberos.KerberosSaslTransportPlugin"
java.security.auth.login.config: "/path/to/jaas.conf"
nimbus.authorizer: "org.apache.storm.security.auth.authorizer.
SimpleACLAuthorizer"
storm.principal.tolocal: "org.apache.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
org.apache.storm.security.auth.authorizer. SimpleACLAuthorizer	Default authorizer for the Nimbus node and all Storm nodes except DRPC.
org.apache.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 `org.apache.storm.security.auth.kerberos.AutoTGT`.
drpc.authorizer	Set to `org.apache.storm.security.auth.authorizer.DRPCSimpleACLAu` to enable authorizer for Storm DRPC node.
nimbus.authorizer:	Set to `org.apache.storm.security.auth.authorizer.SimpleACLAuthor` to enable authorizer for Storm nimbus node.
storm.principal.tolocal:	Set to `org.apache.storm.security.auth.KerberosPrincipalToLocal` to enable transforming kerberos principal to local user names.

Configuration Property	Description
storm.zookeeper.superACL:	Set to `sasl:storm` to 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
     ```

   - Ubuntu or Debian

     ```
     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
     ```

   - Ubuntu

     ```
     service supervisor status
     ```

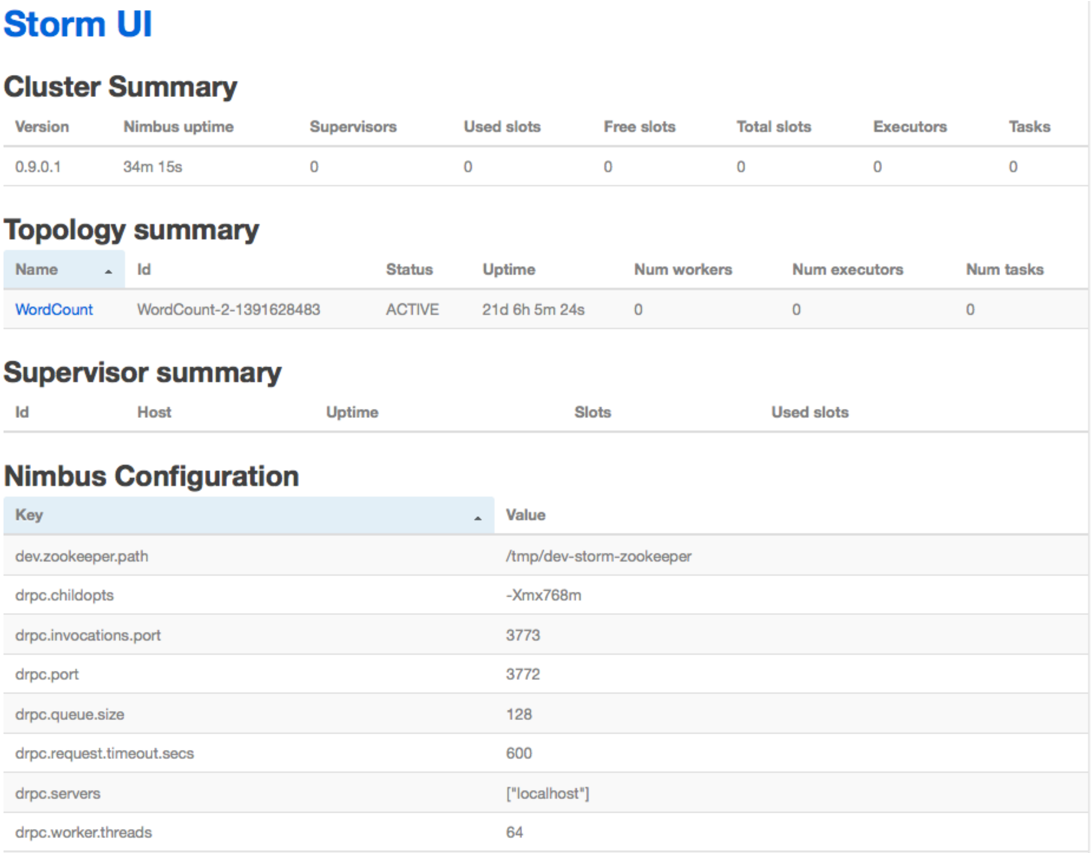
     You should see output similar to the following:

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

3. Point your browser to the following URL:

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

   You should see 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-
topologies-*.jar org.apache.storm.starter.WordCountTopology wordcount
```

19. Installing and Configuring Apache Spark

This section describes how to install and configure Apache Spark for HDP:

- Spark Prerequisites
- Installing Spark
- Configuring Spark
- (Optional) Starting the Spark Thrift Server
- (Optional) Configuring Dynamic Resource Allocation
- Validating Spark

For more information about Spark on HDP (including how to install Spark using Ambari), see the Spark Guide.

19.1. Spark Prerequisites

Before installing Spark, make sure your cluster meets the following prerequisites.

Table 19.1. Prerequisites for running Spark 1.6

Prerequisite	Description
Cluster Stack Version	• HDP 2.4.0 or later
(Optional) Ambari Version	• Ambari 2.2.1 or later
Software dependencies	• Spark requires HDFS and YARN • PySpark requires Python to be installed on all nodes

Note

When you install HDP 2.5.0, Spark 1.6.2 is installed.

19.2. Installing Spark

When you install Spark, the following directories are 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
- `/usr/hdp/current/spark-thriftserver` for the Spark Thrift Server

To install Spark:

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 shows all the versions of Spark available. For example:

```
spark_<version>_<build>-master.noarch : Server for Spark master
spark_<version>_<build>-python.noarch : Python client for Spark
spark_<version>_<build>-worker.noarch : Server for Spark worker
spark_<version>_<build>.noarch : Lightning-Fast Cluster Computing
```

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
     ```

3. Before you launch the Spark Shell or Thrift Server, make sure that you set `$JAVA_HOME`:

   ```
   export JAVA_HOME=<path to JDK 1.8>
   ```

19.3. Configuring Spark

To configure Spark, edit the following configuration files on all nodes that 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`
- `spark-thrift-sparkconf.conf`

Note

Note: the following instructions are for a non-Kerberized cluster.

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=2.5.0.0-1245
```

hive-site.xml

If you plan to use Hive with Spark, create a `hive-site.xml` file in the Spark client `SPARK_HOME/conf` directory. (Note: if you installed the Spark tech preview you can skip this step.)

Edit the file so that it contains only the `hive.metastore.uris` property. Make sure that the `hostname` points to the URI where the Hive Metastore is running.

Important

`hive-site.xml` contains a number of properties that are not relevant to or supported by the Spark thrift server. Ensure that your Spark `hive-site.xml` file contains only the following configuration property.

For example:

```
<property>
     <name>hive.metastore.uris</name>
     <value>thrift://c6401.ambari.apache.org:9083</value>
     <description>URI for client contact metastore server</description>
</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 minimizes 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. For example:

  ```
  spark.yarn.historyServer.address c6401.ambari.apache.org:18080
  spark.history.ui.port 18080
  spark.eventLog.dir hdfs:///spark-history
  spark.eventLog.enabled true
  spark.history.fs.logDirectory hdfs:///spark-history
  spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider
  ```

- Delete the `spark.yarn.services` property, if specified in the file.

If you submit jobs programmatically in a way that `spark-env.sh` is not executed during the submit step, or if you wish to specify a different cluster version than the version installed on the client, set the following two additional property values:

```
spark.driver.extraJavaOptions -Dhdp.version=<HDP-version>
spark.yarn.am.extraJavaOptions -Dhdp.version=<HDP-version>
```

For example:

```
spark.driver.extraJavaOptions -Dhdp.version=2.5.0.0-1245
spark.yarn.am.extraJavaOptions -Dhdp.version=2.5.0.0-1245
```

spark-thrift-sparkconf.conf

Add the following properties and values to the `spark-thrift-sparkconf.conf` file:

```
spark.eventLog.dir hdfs:///spark-history
spark.eventLog.enabled true
spark.history.fs.logDirectory hdfs:///spark-history
spark.history.provider org.apache.spark.deploy.history.FsHistoryProvider
```

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
```

Create an HDFS Directory

As the hdfs service user, create an HDFS directory called spark-history with user:spark, user group:hadoop, and permissions = 777:

```
hdfs dfs -mkdir /spark-history
hdfs dfs -chown -R spark:hadoop /spark-history
hdfs dfs -chmod -R 777 /spark-history
```

19.4. (Optional) Starting the Spark Thrift Server

To enable and start the Spark Thrift Server:

1. From SPARK_HOME, start the Spark SQL Thrift Server. Specify the port value of the Thrift Server (the default is 10015). For example:

   ```
   su spark

   ./sbin/start-thriftserver.sh --master yarn-client --executor-
   memory 512m --hiveconf hive.server2.thrift.port=100015
   ```

2. Use this port when you connect via Beeline.

Kerberos Considerations

If you are installing the Spark Thrift Server on a Kerberos-secured cluster, the following instructions apply:

- The Spark Thrift Server must run in the same host as `HiveServer2`, so that it can access the `hiveserver2` keytab.
- Edit permissions in `/var/run/spark` and `/var/log/spark` to specify read/write permissions to the Hive service account.
- Use the Hive service account to start the `thriftserver` process.

Note

We recommend that you run the Spark Thrift Server as user `hive` instead of user `spark` (this supersedes recommendations in previous releases). This ensures that the Spark Thrift Server can access Hive keytabs, the Hive metastore, and data in HDFS that is stored under user `hive`.

Important

When the Spark Thrift Server runs queries as user `hive`, all data accessible to user `hive` is accessible to the user submitting the query. For a more secure configuration, use a different service account for the Spark Thrift Server. Provide appropriate access to the Hive keytabs and the Hive metastore.

For Spark jobs that are not submitted through the Thrift Server, the user submitting the job must have access to the Hive metastore in secure mode (via kinit).

19.5. (Optional) Configuring Dynamic Resource Allocation

The dynamic resource allocation feature allocates resources as you need them and releases them when you do not need them, rather than reserving the resources.

Note

If you are using the Spark Thrift Server, there is no need to set up dynamic resource allocation. As of HDP 2.4.2, the Spark Thrift Server automatically uses dynamic resource allocation. You can also enable dynamic resource allocation for individual Spark jobs. For more information, see Configuring Dynamic Resource Allocation in the *Spark Guide*.

19.6. Validating Spark

To validate the Spark installation, run the Spark jobs described in the Running Spark Applications section of the *Spark 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
2. Configure Kafka
3. Validate Kafka

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:
  - EXT4: recommended
  - EXT3: supported

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
     ```

   - For Ubuntu and Debian

     ```
     apt-get 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
log.dirs	Comma-separated list of directories where Kafka log files are stored. The default value is /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 on which we listen 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,SASL_PLAINTEXT://:9091`
	`PLAINTEXT://0.0.0.0:9092, SASL_PLAINTEXT://localhost:9093`

File descriptor	Kafka uses a very large number of files and also large number of sockets to communicate with clients. We suggest the following values: • The maximum number of open files. Recommended value: 128000

Kafka Configuration Property	Description
	• The maximum number of processes. Recommended value: 65536

20.3. Validate Kafka

Use the following procedure to verify the Kafka installation and configuration.

Before you begin:

- Verify that Zookeeper is running before starting Kafka and validating the installation.
- Set KAFKA_HOME to /usr/hdp/current/kafka-broker.

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 occurs. 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 and Configuring Zeppelin

This section describes how to install and configure Apache Zeppelin for Hortonworks Data Platform (HDP):

- Installation Prerequisites
- Installing the Zeppelin Package
- Configuring Zeppelin
- Starting, Stopping, and Restarting Zeppelin
- Validating Zeppelin
- Accessing the Zeppelin UI

For more information about Zeppelin on HDP , see the Zeppelin Guide

21.1. Installation Prerequisites

Before installing Zeppelin, ensure that your cluster meets the following prerequisites.

Table 21.1. Installation Prerequisites

Prerequisite	Description
Cluster stack version	HDP 2.5.0 or later
Software dependencies	Required: • HDFS • Apache Spark • YARN • Apache ZooKeeper Optional: • Apache Hive • Livy

21.2. Installing the Zeppelin Package

When you install Zeppelin, the following directories are created:

`/etc/zeppelin/conf`	Configuration file for submitting Spark jobs
`/usr/hdp/current/zeppelin-server`	Zeppelin server
`/var/lib/zeppelin`	Sample notebooks

`/usr/hdp/current/zeppelin-server/notebook`	Zeppelin logs
`/var/run/zeppelin`	Zeppelin PID directory

Follow these steps to install the Zeppelin package:

1. Search for Zeppelin in the HDP repository:

    - Use this command if you are using RHEL or CentOS:

      yum search zeppelin

    - Use this command if you are using SLES:

      zypper install zeppelin

    - Use this command if you are using Ubuntu or Debian:

      apt-cache zeppelin

    The following example shows all the available versions of Zeppelin:

    ```
    zeppelin.noarch : zeppelin Distro virtual package
    zeppelin_<version>_<build>.noarch : Web-based notebook for Apache Zeppelin
    ```

2. Install the Zeppelin version that corresponds to the HDP version that you currently have installed:

    - Use this command if you are using RHEL or CentOS:

      ```
      yum install zeppelin_<version>
      ```

    - Use this command if you are using SLES:

      ```
      zypper install zeppelin_<version>
      ```

    - Use this command if you are using Ubuntu or Debian:

      ```
      apt-get install zeppelin_<version>
      ```

21.3. Configuring Zeppelin

Perform the following steps to configure Zeppelin:

1. There are two ways to set environmental variables in Zeppelin.

    a. You can edit the `/usr/hdp/current/zeppelin-server/conf/zeppelin-env.sh` file to set the following environmental variables:

    ```
    export JAVA_HOME=<path to JDK 1.8>
    export PATH=$JAVA_HOME/bin:$PATH
    export SPARK_HOME=/usr/hdp/current/spark-client
    export HADOOP_CONF_DIR=/etc/hadoop/conf
    export ZEPPELIN_PORT=9995
    ```

Environment variables override Java properties.

b. If you do not want to change the environmental variables in /usr/hdp/current/zeppelin-server/conf/zeppelin-env.sh, you can copy /usr/hdp/current/zeppelin-server/conf/zeppelin-site.xml.template to /usr/hdp/current/zeppelin-server/conf/zeppelin-site.xml and add Java properties. See the Apache Apache Zeppelin Quick Start.

2. If you are interested in using Java properties with Zeppelin, see the Apache Apache Zeppelin Quick Start. Environment variables override Java properties.

3. If you are interested in using Shiro authentication for Zeppelin, edit the shiro.ini file as specified in Shiro authentication for Apache Zeppelin.

21.4. Starting, Stopping, and Restarting Zeppelin

Before you start, stop, or restart Zeppelin, you must switch user to zeppelin:

```
su zeppelin
```

Follow this step to start the Zeppelin server:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh start
```

Follow this step to stop the Zeppelin server:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh stop
```

Follow this step to restart the Zeppelin server:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh restart
```

21.5. Validating Zeppelin

After successfully starting the Zeppelin server, you should validate the installation by checking the Zeppelin daemon status and the installed version.

There are two ways to check Zeppelin daemon status. Use the following command to check Zeppelin daemon status from the command line:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh status
```

You can also check the status by opening the Zeppelin host with the port number that you configured for it in zeppelin-env.sh in a web browser: for example, http://zeppelin.local:9995.

Use the following command to check the version number of Zeppelin:

```
/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh --version
```

21.6. Accessing the Zeppelin UI

To access the Zeppelin UI, enter the following address into your browser:

```
http://<zeppelin_host>:<port>
```

In this example, <zeppelin_host> is the node on which you installed Zeppelin, and <port> is the port on which you installed Zeppelin (default port is 9995).

22. 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.

1. Install the Accumulo Package
2. Configure Accumulo
3. Configuring the "Hosts" Files
4. Validate Accumulo
5. Smoke Test Accumulo

22.1. 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
```

- For Ubuntu and Debian:

```
apt-get install accumulo
```

22.2. 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 is able to be the active master, the rest are 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
```

22.3. 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 results in the appropriate process actively running on each of the listed hosts. For the other files, while the appropriate process starts on each listed host, only one of the started processes is active. The other processes stay in a hot-standby state, attempting to become the active process that enables high-availability deployments.

22.4. 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.

22.5. 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
   ```

23. 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.

1. Installing the Falcon Package
2. Setting Directories and Permissions
3. Configuring Proxy Settings
4. Configuring Falcon Entities
5. Configuring Oozie for Falcon
6. Configuring Hive for Falcon
7. Configuring for Secure Clusters
8. Validate Falcon

Note

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

23.1. 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
     ```

   - For Ubuntu or Debian:

     ```
     apt-get 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.

23.2. 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.

23.3. 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.

23.4. 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).

23.5. 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-create</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,
         currentWeek=org.apache.oozie.extensions.
OozieELExtensions#ph2_currentWeek,
         lastWeek=org.apache.oozie.extensions.
OozieELExtensions#ph2_lastWeek,
         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#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-job-submit-data</
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,
         currentWeek=org.apache.oozie.extensions.
OozieELExtensions#ph1_currentWeek_echo,
         lastWeek=org.apache.oozie.extensions.
OozieELExtensions#ph1_lastWeek_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,
         dataIn=org.apache.oozie.extensions.
OozieELExtensions#ph1_dataIn_echo,
         instanceTime=org.apache.oozie.coord.
CoordELFunctions#ph1_coord_nominalTime_echo_wrap,
         formatTime=org.apache.oozie.coord.
CoordELFunctions#ph1_coord_formatTime_echo,
         dateOffset=org.apache.oozie.coord.
CoordELFunctions#ph1_coord_dateOffset_echo,
         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 ensures 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/2.5.0.0-<$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-<
$version>.jar /usr/hdp/2.5.0.0-<$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.

23.6. 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.

23.7. 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>
   ```

23.8. 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
   ```

24. 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
2. Set up and Validate the Knox Gateway Installation
3. Configuring Knox Single Sign-on

24.1. 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
  ```

- For Ubuntu:

  ```
  apt-get 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`

24.2. 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 Apache Know Gateway Overview in the *Hadoop Security Guide*.

To set up the gateway and test access:

1. Set the master secret.

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

   You are prompted for the master secret. Enter the password at the 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 *Hadoop Security 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.

24.3. Configuring Knox Single Sign-on (SSO)

Knox Single Sign-on (SSO) introduces the ability to configure a single username and password for access control to multiple web UIs. This feature leverages the hadoop-auth module in Hadoop common to use a common SSO cookie for web UIs while retaining the non-web browser authentication through kerberos/SPNEGO. To configure the Knox SSO feature, complete the following step.

Configure the following properties in the `knoxsso.xml` file located in `{GATEWAY_HOME}/conf/topologies`.

Parameter	Description	Default
knoxsso.cookie.secure.only	This determines whether the browser is allowed to send the cookie over unsecured channels. This should always be set to true in production systems. If during development a relying party is not running ssl then you can turn this off. Running with it off exposes the cookie and underlying token for capture and replay by others.	true
knoxsso.cookie.max.age	optional: This indicates that a cookie can only live for a specified amount of time - in seconds. This should probably be left to the default which makes it a session cookie. Session cookies are discarded once the browser session is closed.	session
knoxsso.cookie.domain.suffix	optional: This indicates the portion of the request hostname that represents the domain to be used for the cookie domain. For single host development scenarios the default behavior should be fine. For production deployments, the expected domain should be set and all configured URLs that are related to SSO should use this domain. Otherwise, the cookie is not presented by the browser to mismatched URLs.	Default cookie domain or a domain derived from a hostname that includes more than 2 dots.
knoxsso.token.ttl	This indicates the lifespan of the token within the cookie. Once it expires a new cookie must be acquired from KnoxSSO. This is in milliseconds. The 36000000 in the topology above gives you 10 hrs.	30000 That is 30 seconds.
knoxsso.token.ttl	This is a comma separated list of audiences to add to the JWT token. This is used to ensure that a token received by a participating application knows that the token was intended for use with that application. It is optional. In the event that an application has expected audiences and they are not present the token must be rejected. In the event where the token has audiences and the application has none expected then the token is accepted. OPEN ISSUE - not currently being populated in WebSSOResource.	empty
knoxsso.redirect.whitelist.regex	A semicolon separated list of regex expressions. The incoming originalUrl must match one of the expressions in order for KnoxSSO to redirect to it after authentication. Defaults to only relative paths and localhost with or without SSL for development usecases. This needs to be opened up for production use and actual participating applications. Note	^/.*$;^https?://localhost:\d{0,9}/.*$

Parameter	Description	Default
	that cookie use is still constrained to redirect destinations in the same domain as the KnoxSSO service - regardless of the expressions specified here.	

The following is a sample KnoxSSO topology.

```
<topology>
      <gateway>
          <provider>
              <role>authentication</role>
              <name>ShiroProvider</name>
              <enabled>true</enabled>
              <param>
                  <name>sessionTimeout</name>
                  <value>30</value>
              </param>
              <param>
                  <name>main.ldapRealm</name>
                  <value>org.apache.hadoop.gateway.shirorealm.KnoxLdapRealm</
value>
              </param>
              <param>
                  <name>main.ldapContextFactory</name>
                  <value>org.apache.hadoop.gateway.shirorealm.
KnoxLdapContextFactory</value>
              </param>
              <param>
                  <name>main.ldapRealm.contextFactory</name>
                  <value>$ldapContextFactory</value>
              </param>
              <param>
                  <name>main.ldapRealm.userDnTemplate</name>
                  <value>uid={0},ou=people,dc=hadoop,dc=apache,dc=org</value>
              </param>
              <param>
                  <name>main.ldapRealm.contextFactory.url</name>
                  <value>ldap://localhost:33389</value>
              </param>
              <param>
                  <name>main.ldapRealm.contextFactory.
authenticationMechanism</name>
                  <value>simple</value>
              </param>
              <param>
                  <name>urls./**</name>
                  <value>authcBasic</value>
              </param>
          </provider>
          <provider>
              <role>identity-assertion</role>
              <name>Default</name>
              <enabled>true</enabled>
          </provider>
      </gateway>
      <service>
          <role>KNOXSSO</role>
          <param>
```

```
            <name>knoxsso.cookie.secure.only</name>
            <value>true</value>
         </param>
         <param>
            <name>knoxsso.token.ttl</name>
            <value>100000</value>
         </param>
         <param>
            <name>knoxsso.redirect.whitelist.regex</name>
            <value>^/.*$;https?://localhost*$</value>
         </param>
         <param>
            <name>knoxsso.cookie.domain.suffix</name>
            <value>.novalocal</value>
         </param>
     </service>
 </topology>
```

This topology results in a KnoxSSO URL that looks something like:

```
https://{gateway_host}:{gateway_port}/gateway/knoxsso/api/v1/websso
```

25. 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*
     ```

   - For Ubuntu:

     ```
     apt-get 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.
   0.2.5.0.0 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 2.5.0.0-1245
   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]#
```

7. If HDP is installed in a cluster without using Ambari, invoke the following command (as user hdfs) after the slider client is installed:

   ```
   slider dependency --upload
   ```

   The above command uploads the Slider dependency tarball in an HDP specific path:

   ```
   /hdp/apps/<hdp_version>/slider/slider.tar.gz
   ```

   All subsequent Slider application creation jobs use this dependency tarball. The benefit is twofold. The application submission time reduces drastically from approximately 10-20 seconds to approximately 1-2 seconds, depending on the hardware. It also eliminates all potential jar conflict issues when 3rd party client applications use the slider-core jar to submit Slider application creation requests.

26. Installing and Configuring Apache Atlas

This section describes how to install and configure Apache Atlas for HDP:

- Atlas Prerequisites
- Installing Atlas
- Installing Atlas Metadata Apache Hive Plugin
- Configuring Apache Hive Hook
- Configuring Atlas for Apache Kafka Notifications
- Configuring Apache Storm Hook
- Configuring the Graph Database to Use HBase and SolrCloud
- Configuring Apache Falcon Hook
- Configuring Apache Sqoop Hook
- Configuring for Secure Clusters
- Configuring Atlas in a Kerberized Cluster
- Validating Atlas

26.1. Atlas Prerequisites

Before installing Atlas, make sure your cluster meets the following prerequisites:

Table 26.1. Atlas Cluster Prerequisites

Item	Prerequisite
Cluster Stack Version	HDP 2.4.0 or later

26.2. Installing Atlas

To install Atlas, run the following commands as root:

- For RHEL or CentOS:

```
yum install atlas-metadata_2_4_*
```

- For SLES:

```
zypper install atlas-metadata_2_4_*
```

- For Ubuntu and Debian:

  ```
  apt-get install atlas-metadata_2_4_*
  ```

26.3. Installing Atlas Metadata Apache 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_4_*-hive-plugin*
  ```

- For SLES:

  ```
  zypper install atlas-metadata_2_4_*-hive-plugin*
  ```

- For Ubuntu and Debian:

  ```
  apt-get install atlas-metadata_2_4_*-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.

26.4. Configuring Apache 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>
```

4. Add `export HIVE_AUX_JARS_PATH=<atlas package>/hook/hive` to `hive-env.sh` of your Hive configuration.

5. Copy `<atlas-conf>/client.properties` and `<atlas-conf>/application.properties` to the hive conf directory.

6. If Hive actions are executed with Oozie, add the JAR files that are included with the Atlas server to the `/user/oozie/share/lib/lib_$DATE/hive` directory, where $DATE represents the name of the Oozie directory. You can obtain the name of the Oozie directory by executing the command: `hdfs dfs -ls /user/oozie/share/lib`.

   To populate the Oozie directory with the Atlas server JAR files, run the following commands from the Atlas Hive package installation directory:

```
hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/atlas-
client-*.jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/atlas-
typesystem*.jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/hook/hive/hive-
bridge*jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/json4s-
ast*.jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/json4s-
core*jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/json4s-
native*.jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/scala-
compiler*.jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/scala-
library*.jar
/user/oozie/share/lib/lib_$DATE/hive/
```

```
hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/scalap*.
jar
/user/oozie/share/lib/lib_$DATE/hive/

hdfs dfs -copyFromLocal . /usr/hdp/current/atlas-server/bridge/hive/scala-
reflect*.jar
/user/oozie/share/lib/lib_$DATE/hive/
```

26.5. Configuring Atlas For Apache Kafka Notifications

To generate Apache Kafka notifications, you must modify certain Atlas properties. This change is required for Ranger integration or anytime there is a consumer of entity change notifications.

1. Modify the following Atlas properties by adding the prefix 'atlas.kafka'. These properties are located in the `conf/application.properties` file in the ranger-tagsync installation directory:

   Received from Tom's doc:

   ```
   atlas.notification.embedded=true
   atlas.kafka.data=${sys:atlas.home}/data/kafka
   atlas.kafka.zookeeper.connect=localhost:9026
   atlas.kafka.bootstrap.servers=localhost:9027
   atlas.kafka.zookeeper.session.timeout.ms=400
   atlas.kafka.zookeeper.sync.time.ms=20
   atlas.kafka.auto.commit.interval.ms=1000
   atlas.kafka.hook.group.id=atlas
   ```

   Note

   For “atlas.kafka.bootstrap.servers” and “atlas.kafka.zookeeper.connect”, modify host names based on your cluster topology. The directory specified in “atlas.kafka.data” should exist.

2. Add an export of HADOOP_CLASSPATH which includes the required Atlas directories to the hive-env configuration export `HADOOP_CLASSPATH=/etc/atlas/conf:/usr/hdp/current/atlas-server/hook/hive:${HADOOP_CLASSPATH}`.

   Note

   It is important that the Atlas directories are prepended to the existing classpath.

3. Restart the Atlas and Hive services after the cluster is fully provisioned.

26.6. Configuring Apache Storm Hook

The goal of integrating Apache Storm with Apache Atlas is to push the operational topology metadata, along with the underlying data source(s), target(s), derivation processes, and any available business context so Atlas can capture the lineage for this topology. Complete the following instructions to integrate Apache Storm with Apache Atlas.

1. Configure the Atlas hook class as `storm.topology.submission.notifier.plugin.class` in the `storm.yaml` file (`/etc/storm/conf/storm.yaml`). For example:

   ```
   storm.topology.submission.notifier.plugin.class:"
   org.apache.atlas.hooks.StormAtlasHook"
   ```

2. Make sure you installed Atlas hook jars in `$<STORM_HOME>/extlib`.

   Where <STORM_HOME> is the Storm installation path.

3. Replace STORM_HOME with the storm installation path.

4. Restart all daemons after you installed the atlas hook jar.

26.7. Configuring the Graph Database

Atlas uses Titan as its its metadata storage and indexing repository. Titan supports BerkeleyDB and HBase as its storage backend. It supports ElasticSearch (embedded) and Solr as its indexing backend. For more information, see Choosing Between Storage Backends and Choosing Between Indexing Backends below.

All Titan configurations are set in the `application.properties` file, located in the `/conf` directory at the location where Atlas is deployed. The `application.properties` file contains sections where you can set the following:

- Set HBase as the backend graph persistence engine
- Set SolrCloud as the backend indexing engine

26.7.1. Choosing Between Storage Backends

BerkeleyDB is suitable for smaller data sets up to 10 million vertices with ACID guarantees. It can be used for proofs of concept or testing in developer environments. On the other hand, HBase does not provide ACID guarantees, but is capable of scaling for large graphs. In addition, HBase inherently provides high availability and generally has higher expected response times while creating types and entities than does BerkeleyDB. For further details, see http://s3.thinkaurelius.com/docs/titan/0.5.4/bdb.html and http://s3.thinkaurelius.com/docs/titan/0.5.4/hbase.html.

26.7.2. Choosing Between Indexing Backends

By default, Atlas runs ElasticSearch in embedded mode as its indexing backend, which works well for proofs of concept or testing in developer environments. For production environments, Hortonworks recommends using Solr in cloud mode for HDP clusters. For further details, see http://s3.thinkaurelius.com/docs/titan/0.5.4/elasticsearch.html and http://s3.thinkaurelius.com/docs/titan/0.5.4/solr.html.

26.7.3. Configure Atlas to Use Apache HBase

Prerequisites for Switching to HBase as the Storage Backend for the Graph Repository

- **Supported HBase versions:**Currently, only HBase versions 1.1.x are supported
- **To run HBase as a distributed cluster:** You must have:
  - 3 or 5 ZooKeeper nodes
  - At least 3 RegionServer nodes. It is ideal to run the DataNodes on the same hosts as the RegionServers for data locality
- **Clear the data in the indexing backend:** Before you switch to HBase as your storage backend, you must clear the data in the indexing backend. If you do not, there might be discrepancies between the storage and indexing backends, which can result in errors during searches.

  ElasticSearch runs by default in embedded mode, so you can clear the data by deleting the `ATLAS_HOME/data/es` directory.

  To clear Solr data, delete the collections that are created during installation of Atlas: `vertex_index`, `edge_index`, and `fulltext_index`. This cleans up the indexing data.
- To configure Atlas to use HBase on an unsecured cluster [245]
- To configure Atlas to use HBase on a cluster secured by Kerberos [246]

To configure Atlas to use HBase on an unsecured cluster:

1. In the graph persistence engine section of the `application.properties` file, set the following properties:

```
#Set the backend graph persistence engine to HBase
atlas.graph.storage.backend=hbase

#For standalone mode, specify 'localhost' as the hostname; for distributed
 mode, specify the ZooKeeper quorum as the hostname
atlas.graph.storage.hostname=<localhost|ZooKeeper_quorum>
```

Note

The value for *`ZooKeeper_quorum`* can be retrieved from the `hbase.zookeeper.quorum` property setting in the `hbase-site.xml` file. For example:

```
<property>
     <name>hbase.zookeeper.quorum</name>
     <value>c6401.ambari.apache.org,c6402.ambari.apache.org,
c6403.ambari.apache.org</value>
</property>
```

2. Add `HBASE_CONF_DIR` to the classpath so the Titan graph database can retrieve the client configuration settings from the `hbase-site.xml` file:

```
export HBASE_CONF_DIR=/etc/hbase/conf
```

3. Ensure that `hbase-site.xml` includes a value for the `zookeeper.znode` parent property. For example:

   ```
   <property>
   <name>zookeeper.znode.parent</name>
   <value>/hbase-unsecure</value>
   </property>
   ```

4. Restart Atlas.

   ```
   /usr/hdp/current/atlas-server/bin/atlas_stop.py
   /usr/hdp/current/atlas-server/bin/atlas_start.py
   ```

To configure Atlas to use HBase on a cluster secured by Kerberos:

Prerequisite: Setting Permissions

When Atlas is configured to use HBase as the storage backend, the Titan graph database needs sufficient user permissions to create and access an HBase table. In a secure cluster, it might be necessary to grant permissions to the `atlas` user for the Titan table.

You can use Apache Ranger to configure a policy for the Titan database. See Configuring Resource-Based Services in the *Hadoop Security Guide*.

If you do not use Apache Ranger, the HBase shell can be used to set the permissions:

```
su hbase
kinit -k -t <HBase_keytab> <HBase_principal>
echo "grant 'atlas', 'RWXCA', 'titan'" | hbase shell
```

1. Create an Atlas JAAS (Java Authentication and Authorization Service) configuration file at the following location:

   ```
   /etc/atlas/conf/atlas-jaas.conf
   ```

   Add configurations into the `atlas-jaas.conf` file that point to your keyTab and principal. For example:

   ```
   Client {
      com.sun.security.auth.module.Krb5LoginModule required
      useKeyTab=true
      useTicketCache=false
      keyTab="/etc/security/keytabs/atlas.service.keytab"
      principal="atlas/c6401.ambari.apache.org@EXAMPLE.COM";
   };
   ```

2. In the Atlas environment variable `METADATA_OPTS`, set `java.security.auth.login.config` to point to the JAAS configuration file that you created in Step 1.

3. Update the `hbase-site.xml` file to include the following configurations:

   ```
   <property>
        <name>zookeeper.znode.parent</name>
        <value>/hbase-secure</value>
   <property>
   ```

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

<property>
     <name>hbase.security.authorization</name>
     <value>true</value>
</property>

<property>
     <name>hbase.master.kerberos.principal</name>
     <value>hbase/_HOST@EXAMPLE.COM</value>
</property>

<property>
     <name>hbase.master.keytab.file</name>
     <value>/etc/security/keytabs/hbase.service.keytab</value>
</property>

<property>
     <name>hbase.regionserver.kerberos.principal</name>
     <value>hbase/_HOST@EXAMPLE.COM</value>
</property>

<property>
     <name>hbase.regionserver.keytab.file</name>
     <value>/etc/security/keytabs/hbase.service.keytab</value>
</property>
```

4. Add `HBASE_CONF_DIR` to the classpath so the Titan graph database can retrieve the client configuration settings from the `hbase-site.xml` file:

```
export HBASE_CONF_DIR=/etc/hbase/conf
```

5. Create `hbase_master_jaas.conf` and `hbase_regionserver_jaas.conf` configuration files at the following locations:

```
/usr/hdp/current/hbase-regionserver/conf/hbase_master_jaas.conf
/usr/hdp/current/hbase-regionserver/conf/hbase_regionserver_jaas.conf
```

   Add configurations into these files that point to your keyTab and principal. For example:

```
Client {
   com.sun.security.auth.module.Krb5LoginModule required
   useKeyTab=true
   storeKey=true
   useTicketCache=false
   keyTab="/etc/security/keytabs/hbase.service.keytab"
   principal="hbase/c6401.ambari.apache.org@EXAMPLE.COM";
};
```

6. Update the `hbase-env.sh` script so that the environment variables HBASE_MASTER_OPTS and HBASE_REGIONSERVER_OPTS include `java.security.auth.login.config` for the `jaas.conf` files that you created in Step 1 and Step 5.

   For example:

```
export
   HBASE_MASTER_OPTS="-Djava.security.auth.login.config=/usr/hdp/current/
hbase-regionserver/conf/hbase_master_jaas.conf …"

export
   HBASE_REGIONSERVER_OPTS="-Djava.security.auth.login.config=/usr/hdp/
current/hbase-regionserver/conf/hbase_regionserver_jaas.conf …”
```

7. Restart the services.

26.7.4. Configure Atlas to Use SolrCloud

Important

Currently, Atlas supports Solr in cloud mode only. HTTP mode is not supported. For more information, refer to the Solr documentation at https://cwiki.apache.org/confluence/display/solr/SolrCloud.

Prerequisites for Switching to SolrCloud as the Indexing Backend for Atlas

- **Memory:** Solr is both memory and CPU intensive. Ensure that the server that is running Solr has adequate memory, CPU, and disk space. Solr works well with 32GB of RAM. Provide as much memory as possible to the Solr process.

- **Disk:** If you must store a large number of entities, make sure you have at least 500 GB of free space in the volume where Solr stores the index data.

- **SolrCloud support for replication and sharding:** Hortonworks recommends that you use SolrCloud with at least 2 Solr nodes running on different servers with replication enabled. ZooKeeper must be installed and configured with 3 to 5 ZooKeeper nodes for SolrCloud.

- **Clear the storage backend data:** When you switch the indexing backend, you must clear the storage backend data. Otherwise, there might be discrepancies between the storage and indexing backends because when you switch the indexing backend, indexing data is lost. This might cause fulltext queries to not work properly on the existing data.

  To clear data for BerkeleyDB, delete the `ATLAS_HOME/data/berkeley` directory.

  To clear data for HBase, in the HBase shell, run:

  ```
  disable titan
  drop titan
  ```

To configure the Titan graph database to use SolrCloud as the indexing backend on unsecured clusters:

1. Download and install Solr version 5.2.1. You can download Solr 5.2.1 from here. Documentation is included in the download.

2. Start Solr in cloud mode.

   SolrCloud mode uses a ZooKeeper service as a central location for cluster management. For a small cluster, you can use an existing ZooKeeper quorum. For larger clusters, use a separate ZooKeeper quorum with at least 3 servers.

3. If the Atlas instance and the Solr instance are located on 2 different hosts, copy the required configuration files from the `ATLAS_HOME/conf/solr` directory on the Atlas instance host to the Solr instance host. Then run the following commands from the SOLR_HOME directory to create collections in Solr that correspond to the indexes that Atlas uses:

   ```
   bin/solr create -c vertex_index -d SOLR_CONF -shards <#numShards> -
   replicationFactor <#replicationFactors>
   bin/solr create -c edge_index -d SOLR_CONF -shards <#numShards> -
   replicationFactor <#replicationFactors>
   bin/solr create -c fulltext_index -d SOLR_CONF -shards <#numShards> -
   replicationFactor <#replicationFactors>
   ```

   Where `SOLR_CONF` refers to the directory where the Solr configuration files have been copied to on Solr host from the Atlas host.

   If `numShards` and `replicationFactor` are not specified, they default to 1, which is adequate if you are testing Solr with Atlas on a single node instance. Otherwise, specify the number of hosts that are in the Solr cluster and the `maxShardsPerNode` configuration.

   Important

   The number of shards cannot exceed the total number of Solr nodes in your SolrCloud cluster.

4. Set the following Atlas configuration parameters in the `application.properties` file that is located in the `/conf` directory:

   ```
   atlas.graph.index.search.backend=solr5
   atlas.graph.index.search.solr.mode=cloud
   atlas.graph.index.search.solr.zookeeper-url=<ZooKeeper_quorum>
   ```

   Where *`ZooKeeper_quorum`* is the ZooKeeper quorum that is set up for Solr as a comma-separated value. For example: `10.1.6.4:2181,10.1.6.5:2181`

5. Restart Atlas.

   ```
   /usr/hdp/current/atlas-server/bin/atlas_stop.py
   /usr/hdp/current/atlas-server/bin/atlas_start.py
   ```

26.8. Configuring Apache Falcon Hook

The Apache Falcon hook sends messages about Falcon entities to the notification server so Atlas can read them and register the entities. To use the Falcon hook with Atlas, add the following property settings to the specified configuration files:

1. Add the following property to the `application.services` section in the `<falcon-conf>/startup.properties` file:

   ```
   org.apache.falcon.atlas.service.AtlasService
   ```

2. Link the Falcon hook JAR files in the FALCON_CLASSPATH:

   ```
   ln -s <ATLAS_HOME>/hook/falcon/* <FALCON_HOME>/server/webapp/falcon/WEb-INF/
   lib/
   ```

3. Copy the `<atlas-conf>/client.properties` file and the `<atlas-conf>/application.properties` file to the `<falcon-conf>` directory.

Limitations

Currently, the Falcon hook has the following limitations:

- Only process entity creation is updated.
- For Falcon cluster entities, the cluster name should be uniform across components that are used, such as Hive, Falcon, and Sqoop. If used with Ambari, the Ambari cluster name should be used for the Falcon cluster entity name.

26.9. Configuring Apache Sqoop Hook

Apache Sqoop has added a `SqoopDataPublisher` class that publishes data to Atlas after import jobs are completed. Today, only hiveImport is supported in sqoopHook. This is used to add entities in Atlas using the model defined in org.apache.atlas.sqoop.model.SqoopDataModelGenerator. Complete the following instructions in your Sqoop set-up to add the Sqoop hook for Atlas in the `<sqoop-conf>/sqoop-site.xml` file:

1. Add the Sqoop job publisher class. Currrently only one publishing class is supported.

```
<property>
     <name>sqoop.job.data.publish.class</name>
     <value>org.apache.atlas.sqoop.hook.SqoopHook</value>
</property>
```

2. Add the Atlas cluster name:

```
<property>
     <name>atlas.cluster.name</name>
     <value><clustername></value>
</property>
```

3. Copy the application and client properties from the Atlas config address.

4. Define `atlas.cluster.name` and `atlas.rest.address` properties in the Sqoop configuration file `sqoop-site.xml` file.

5. Add ATLAS_HOME to the `/usr/hdp/<version>/sqoop/bin`.

```
export ATLAS_HOME=${ATLAS_HOME:-/usr/hdp/2.5.0.0-1245/atlas}
```

6. Add the following information to the `$SQOOP_HOME/bin/configure-sqoop` file after the line `ZOOCFGDIR=${ZOOCFGDIR:-/etc/zookeeper}` .

```
if [ -e "$ATLAS_HOME/hook/sqoop" -a -e "$ATLAS_HOME/hook/hive" ]; then
  for f in $ATLAS_HOME/hook/sqoop/*.jar; do
    SQOOP_CLASSPATH=${SQOOP_CLASSPATH}:$f;
  done
  for f in $ATLAS_HOME/hook/hive/*.jar; do
    SQOOP_CLASSPATH=${SQOOP_CLASSPATH}:$f;
  done
fi
```

7. Copy the Atlas <atlas-conf>/application.properties file and the <atlas-conf>/client.properties file to the <sqoop-conf>/ directory.

8. Link <atlas-home>/hook/sqoop/*.jar in sqoop lib.

Limitations

Currently, only hiveImport jobs are published to Atlas by the Sqoop hook.

26.10. 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>
```

26.11. Configuring Atlas in a Kerberized Cluster

If your cluster is in a Kerberized environment, complete the following steps to configure Atlas.

1. Create the Atlas JAAS configuration file.

   ```
   /etc/atlas/conf/atlas-jass.conf
   ```

2. Update Atlas METADATS_OPTS to include the login configuration. METADATA_OPTS must include "java.security.auth.config' set to the above Atlas JAAS configuration file. For example:

   ```
   -Xmx1024m -Djava.security.auth.login.config=/etc/atlas/conf/
   atlas-jaas.conf
   ```

3. Update the hbase-site.xml file to include the following:

   ```
   <property>
         <name>zookeeper.znode.parent</name>
         <value>/hbase-secure</value>
       </property>

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

       <property>
         <name>hbase.security.authorization</name>
         <value>true</value>
       </property>

       <property>
         <name>hbase.master.kerberos.principal</name>
         <value>hbase/_HOST@EXAMPLE.COM</value>
       </property>

       <property>
         <name>hbase.master.keytab.file</name>
         <value>/etc/security/keytabs/hbase.service.keytab</value>
       </property>

       <property>
         <name>hbase.regionserver.kerberos.principal</name>
         <value>hbase/_HOST@EXAMPLE.COM</value>
       </property>

       <property>
         <name>hbase.regionserver.keytab.file</name>
         <value>/etc/security/keytabs/hbase.service.keytab</value>
   ```

```
    </property>
```

4. Create the `hbase_master_jaas.conf` file and the `hbase_regionserver_jaas.conf` file in `/usr/hdp/current/hbase_regionserver/conf`.

5. Update `hbase-env.sh`. HBASE_MASTER_OPTS and HBASE_REGIONSERVER_OPTS must include ‘java.security.auth.login.config’ for the above jaas.conf files. For example:

```
export
HBASE_MASTER_OPTS="-Djava.security.auth.login.config=/usr/hdp/current/hbase-
regionserver/conf/hbase_master_jaas.conf …”

export
HBASE_REGIONSERVER_OPTS="-Djava.security.auth.login.config=/usr/hdp/current/
hbase-regionserver/conf/hbase_regionserver_jaas.conf …”
```

6. Restart the services.

26.12. 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/
```

27. Setting Up Kerberos Security for Manual Installs

For information on enabling Kerberos security for a manually installed version of HDP, refer to the Hadoop Security Guide:

- Preparing Kerberos
- Configuring HDP
- Setting up One-Way Trust with Active Directory

28. 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
  ```

- For Ubuntu/Debian:

  ```
  sudo apt-get 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
     ```

   - For Ubuntu/Debian:

     ```
     sudo apt-get 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\*
  ```

- For Ubuntu/Debian:

  ```
  sudo apt-get 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\*
      ```

    - For Ubuntu/Debian:

      ```
      sudo apt-get 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\*
      ```

    - For Ubuntu/Debian:

      ```
      sudo apt-get 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\*
      ```

    - For Ubuntu/Debian:

      ```
      sudo apt-get 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\*
  ```

- For Ubuntu/Debian:

  ```
  sudo apt-get 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\*
      ```

    - For Ubuntu/Debian:

      ```
      sudo apt-get 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\*
    ```