Hortonworks DataFlow

Command Line Installation

(Feb 24, 2017)

docs.hortonworks.com

Hortonworks DataFlow: 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 Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, Zookeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

Unlike other providers of platforms built using Apache Hadoop, Hortonworks contributes 100% of our code back to the Apache Software Foundation. The Hortonworks Data Platform is Apache-licensed and completely open source. We sell only expert technical support, training and partner-enablement services. All of our technology is, and will remain free and open source.

Please visit the Hortonworks Data Platform page for more information on Hortonworks technology. For more information on Hortonworks services, please visit either the Support or Training page. Feel free to Contact Us directly to discuss your specific needs.

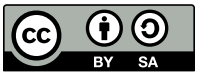


Table of Contents

List of Tables

1. Before You Begin

Before you install HDF, be sure the systems on which you are installing meet the following requirements.

- Supported Operating Systems
- Supported Browsers
- Supported JDKs
- HDP and Component Information
- Hardware Sizing Recommendations
- Downloading HDF

1.1. Supported Operating Systems

- Red Hat Enterprise Linux / CentOS 6 (64-bit)
- Red Hat Enterprise Linux / CentOS 7 (64-bit)
- Ubuntu Precise (12.04) (64-bit)
- Ubuntu Trusty (14.04) (64-bit)
- Debian 7
- SUSE Linux Enterprise Server (SLES) v12 SP1
- SUSE Linux Enterprise Server (SLES) v11 SP4
- SUSE Linux Enterprise Server (SLES) v11 SP3

1.2. Supported Browsers

- Mozilla Firefox: current & current - 1
- Google Chrome: current & current - 1
- Microsoft Edge
- Safari 8

1.3. Supported JDKs

You must have one of the following JDKs installed on the system running HDF.

- Open JDK8

- Oracle JDK 8

1.4. HDP and Component Information

HDF 2.1 includes a number of Apache components, and interoperates with the following versions of HDP and HDP components.

Table 1.1. Supported HDP versions and components

Description	Components
Apache components packaged with HDF 2.1	• Ambari 2.4.2.0 • Kafka 0.10.1 • NiFi 1.1.0 • Ranger 0.6.2 • Storm 1.1.0 • ZooKeeper 3.4.6
Supported HDP versions	• HDP 2.5 • HDP 2.4

Apache component support	• Kafka for GetKafka and PutKafka processor support
	  • Apache Kafka 0.8.x and 0.9.x broker in non-kerberized mode, without SSL
	  • HDP 2.3 and 2.4 Kafka broker in both non-kerberized and kerberized modes, without SSL
	• Kafka for ConsumeKafka and PublishKafka processor support
	  • ConsumeKafka and PublishKafka processor support for the 0.9.0.1 Kafka client
	  • ConsumeKafka_0_10 and PublishKafka_0_10 for the 0.10.x Kafka client
	  •
	  • Apache Kafka 0.9.x and 0.10.x broker in both non-kerberized and kerberized modes, with and without SSL
	  • HDP 2.4 Kafka 0.9.x broker and HDP 2.5 Kafka 0.10.x broker in both non-kerberized and kerberized modes, with and without SSL
	• HDFS
	  • Depends on Hadoop version 2..7.3
	  • Apache HDFS 2.6 and 2.7
	  • HDP 2.4 and 2.5 HDFS in both non-kerberized and kerberized modes
	• HBase
	  • Depends on Hbase version 1.x (Hbase client 1.1.2)
	• Hive
	  • Depends on Apache Hive 1.2.1
	• Storm
	  • Depends on Storm 1.0.1
	  • NiFi Storm spout and bolt work with Storm 1.1.0 in HDP 2.5.

1.5. Hardware Sizing Recommendations

Table 1.2. Hardware sizing recommendations

If you want ...	Recommended hardware sizing ...
50 MB/second sustained throughput and thousands of events per second	• 1 - 2 nodes • 8 or more cores per node, although more is better • 6 or more disks per node (solid state or spinning) • 2 GB memory per node • 1 GB bonded NICs
100 MB/second sustained throughput and tens of thousands of events per second	• 3 - 4 nodes • 8 or more cores per node, although more is better • 6 or more disks per node (solid state or spinning) • 2 GB of memory per node • 1GB bonded NICs
200 MB/second sustained throughput and hundreds of thousands of events per second	• 5 - 7 nodes • 24 or more cores per node (effective CPUs) • 12 or more disks per node (solid state or spinning) • 4 GB of memory per node • 10 GB bonded NICs
400 - 500 MB/second sustained throughput and hundreds of thousands of events per second	• 7 - 10 nodes • 24 or more cores per node (effective CPUs) • 12 or more disks per node (solid state or spinning) • 6 GB of memory per node • 10 GB bonded NICs

1.6. Downloading HDF

Select the installation files appropriate for your OS and operational objectives from the available links in the HDF Release Notes.

2. Installing NiFi

2.1. Installing NiFi

To install NiFi:

1. Extract the files to the location from which you want to run the application.

For information on how to configure NiFi (for instance, security, data storage, or the port on which NiFi is running), see the System Administrator's Guide.

2.1.1. Installing as a Service

You can also install NiFi as a service.

1. Navigate to the NiFi installation directory.
2. Enter:

```
bin/nifi.sh install
```

3. You can specify a custom name by specifying that name during your install command.

   For example, to install NiFi as a service with the name `dataflow`, enter:

```
bin/nifi.sh install dataflow
```

   The NiFi service is installed with the default name `nifi`.

2.2. Starting NiFi

Once you have downloaded and installed HDF, you can start it by using the mechanism appropriate for your operating system.

If you are running NiFi on a Linux or Mac OSX operating system, you can start NiFi to run in the foreground, background, or as a service.

Launching NiFi in the foreground:

1. From a terminal windows, navigate to the NiFi installation directory.
2. Enter:

```
bin/nifi.sh run
```

Launching NiFi in the background:

1. From a terminal window, navigate to the NiFi installation directory.
2. Enter:

```
bin/nifi.sh start
```

Launching NiFi as a service:

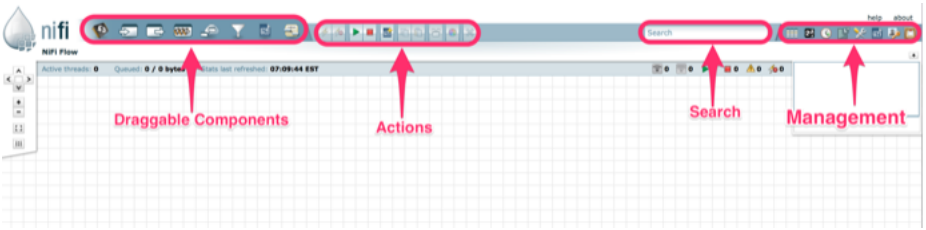
1. From a terminal window, enter:

```
sudo service nifi start
```

2.3. Launching the UI

Once you have started NiFi, you can bring up the User Interface (UI) to create and monitor dataflows. Open a web browser and navigate to `http://localhost:8080/nifi`.

The toolbars at the top of the UI are very important to create your first dataflow:



3. 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

3.1. 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
```

3.2. Securing Zookeeper with Kerberos (optional)

(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
   ```

3.3. 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

3.3.1. 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"
```

3.3.2. 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"
```

3.3.3. 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>
```

3.4. 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**

3.5. 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:

4. ```
   export JAVA_HOME=/usr/jdk64/jdk1.8.0_40
   export ZOOKEEPER_HOME=/usr/hdf/current/zookeeper-server
   export ZOO_LOG_DIR=/var/log/zookeeper
   export ZOOPIDFILE=/var/run/zookeeper/zookeeper_server.pid
   export SERVER_JVMFLAGS=-Xmx1024m
   export JAVA=$JAVA_HOME/bin/java
   export CLASSPATH=$CLASSPATH:$ZOOKEEPER_HOME/*
   ```

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.

3.6. 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/hdf/current/zookeeper-server/
conf ; export ZOOCFG=zoo.cfg; source /usr/hdf/current/zookeeper-server/conf/
zookeeper-env.sh ; /usr/hdf/current/zookeeper-server/bin/zkServer.sh start"
```

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

4. 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

4.1. 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>
   ```

4.2. Configure Kafka

Use the following procedure to configure Kafka.

1. By default, Kafka is installed at `/usr/hdf/current/kafka-broker`.
2. Verify the values of the following configuration properties in the `server.properties` file:

Table 4.1. 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

4.3. 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/hdf/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
   ```

5. 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

5.1. 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
   ```

5.2. 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
   ```

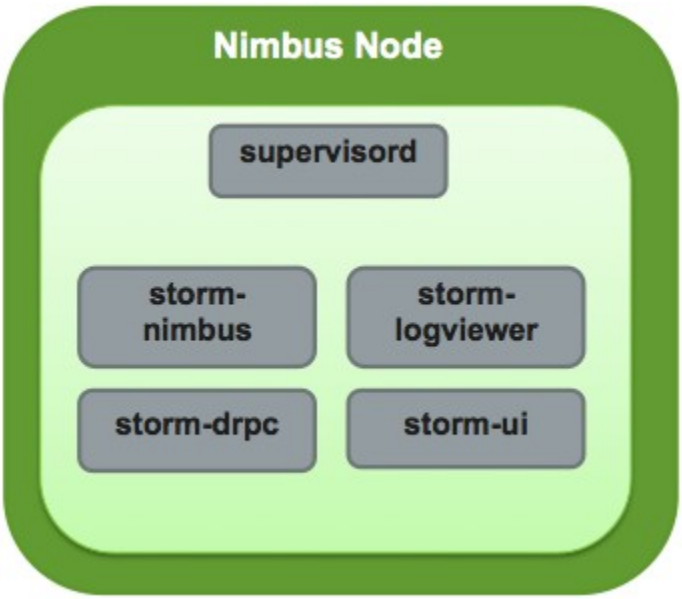
   ```
   chmod -R 755 $STORM_LOCAL_DIR
   ```

5.3. 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
```

5.4. (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 5.1. 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"
```

5.5. (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 5.2. Supported Authorizers

Authorizer	Description
org.apache.storm.security.auth.authorizer. SimpleACLAuthorizer	Default authorizer for the Nimbus node and all Storm nodes except DRPC.

Authorizer	Description
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 5.3. 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.
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/hdf/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 5.4. 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 5.5. 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.

5.6. 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
     ```

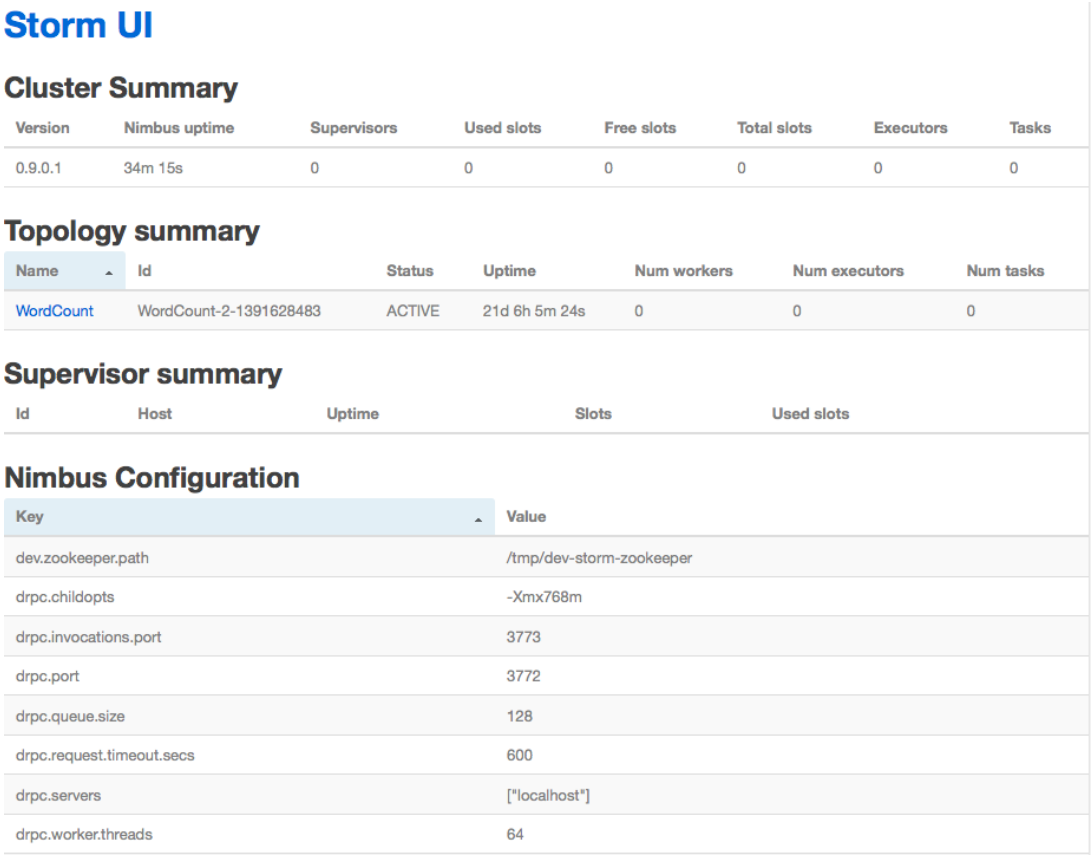
     You should see output similar to the following:

```
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/hdf/current/storm-client/contrib/storm-starter/storm-starter-
topologies-*.jar org.apache.storm.starter.WordCountTopology wordcount
```

6. Installing Apache Ranger

Apache Ranger delivers a comprehensive approach to security for an HDF 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:

- Installing Policy Manager
- Installing UserSync
- Installing Ranger Plug-ins
- Verifying the Installation

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

6.1. 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

6.1.1. Install the Ranger Policy Manager

1. Make sure the HDF 2.0 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/HDF/centos6/2.x/GA/2.0.0.0/
     hdf.repo -O /etc/yum.repos.d/hdf.repo
     ```

   - For Ubuntu

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDF/
     ubuntu<version>/2.x/GA/2.0.0.0/hdf.list -O /etc/apt/sources.list.d/hdf.
     list
     ```

   - For Debian

     ```
     apt-get update wget http://public-repo-1.hortonworks.com/HDF/debian7/2.x/
     GA/2.0.0.0/hdf.list -O /etc/apt/sources.list.d/hdf.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/hdf/<version>/ranger-admin/
```

- Edit the following install.properties entries:

Table 6.1. install.properties Entries

Configuration Property	Default/Example Value	Required?
Ranger Policy Database		
DB_FLAVOR Specifies the type of database used (MYSQL,ORACLE,POSTGRES,MSSQL)	MYSQL (default)	Y
SQL_CONNECTOR_JAR Path to SQL connector jar of the DB Flavor selected. The value should be the absolute path including the jar name.	/usr/share/java/mysql-connector-java.jar (default) /usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	Y
db_root_user database username who has privileges for creating database schemas and users	root (default)	Y
db_root_password database password for the "db_root_user"	rootPassW0Rd	Y
db_host Hostname of the Ranger policy database server	localhost	Y
db_name Ranger Policy database name	ranger (default)	Y
db_user db username used for performing all policy mgmt operation from policy admin tool	rangeradmin (default)	Y
db_password database password for the "db_user"	RangerAdminPassW0Rd	Y
Ranger Audit		
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.

Configuration Property	Default/Example Value	Required?
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
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

Configuration Property	Default/Example Value	Required?
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>
```

6.1.2. Install the Ranger Policy Administration Service

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

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

```
./setup.sh
```

6.1.3. 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".

6.1.4. 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

6.1.5. 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.

6.2. Installing UserSync

In this section:

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

6.2.1. 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.

6.2.1.1. LDAP Connection Check Tool Parameters

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

```
cd /usr/hdf/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/hdf/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.

6.2.1.2. 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.

6.2.1.3. 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>=*

6.2.1.4. 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.

6.2.1.5. 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.

6.2.1.6. Output Directory Content

This tool generates three files in the output directory specified with the `-o` option, or by default to the `/usr/hdf/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.

6.2.1.7. 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.

6.2.1.8. 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"**

6.2.1.9. 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=
```

6.2.2. Install UserSync and Start the Service

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 6.2. 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)
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	cn (default)	N (defaults to cn)

Configuration Property Name	Default/Example Value	Required?
entry that would be treated as user name		
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)
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/hdf/<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.

6.3. Installing Ranger Plug-ins

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

Note

To ensure that you are installing the Hortonworks 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
```

6.3.1. 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/hdf/<version>/ranger-kafka-plugin`.

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

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

Table 6.3. Properties to Edit in the install.properties File

Configuration Property Name	Default/Example Value	Required?
Policy Admin Tool		
COMPONENT_INSTALL_DIR_NAME	/usr/hdf/2.0.0.0-1245/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

Configuration Property Name	Default/Example Value	Required?
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
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/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/hdf/current/kafka-broker/bin/kafka stop"
```

```
su kafka -c "/usr/hdf/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/hdf/2.0.0.0-1245/kafka/config/server.properties`. The value of the authorizer.class.name should be org.apache.ranger.authorization.kafka.authorizer.RangerKafkaAuthorizer.

6.3.2. 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/hdf/<version>/ranger-storm-plugin
   ```

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

Table 6.4. 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	/usr/share/java/mysql-connector-java.jar (default)	Y

Configuration Property Name	Default/Example Value	Mandatory?
absolute path including the jar name.	/usr/share/java/postgresql.jar /usr/share/java/sqljdbc4.jar /usr/share/java/ojdbc6.jar	
XAAUDIT.DB.IS_ENABLED Enable or disable database audit logging. **Note**: If this property is set to FALSE, Ranger will not store audit logs in the audit DB, and audit logs will not be visible in the Ranger UI. If you would like to access audit logs from the UI, set this value to TRUE.	false (default) true	Y
XAAUDIT.DB.FLAVOUR Specifies the type of database used for audit logging (MYSQL,ORACLE, PostgreSQL, 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
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	changeit (default)	If SSL is enabled

Configuration Property Name	Default/Example Value	Mandatory?
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.		

3. Enable the Storm plug-in by entering the following commands:

```
cd /usr/hdf/<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.

6.4. 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