Hortonworks Data Platform

Command Line Upgrade

(June 1, 2017)

docs.hortonworks.com

Hortonworks Data Platform: Command Line Upgrade

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Software Foundation projects that focus on the storage and processing of Big Data, along with operations, security, and governance for the resulting system. This includes Apache Hadoop – which includes MapReduce, Hadoop Distributed File System (HDFS), and Yet Another Resource Negotiator (YARN) – along with Ambari, Falcon, Flume, HBase, Hive, Kafka, Knox, Oozie, Phoenix, Pig, Ranger, Slider, Spark, Sqoop, Storm, Tez, and ZooKeeper. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

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

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

Table of Contents

List of Tables

1. Upgrade from HDP 2.5 to HDP 2.6.0 Manually

This chapter provides instructions on how to manually upgrade to HDP 2.6.0 from the HDP 2.5 release. It assumes the existing HDP 2.5.x was also installed manually.

Important

If you installed and manage HDP 2.5 with Ambari, **you must use the Ambari Upgrade Guide** to perform the HDP 2.5 to HDP 2.6.0 upgrade.

These instructions cover the upgrade between two minor releases. If you need to upgrade between two maintenance releases, follow the upgrade instructions in the HDP Release Notes.

Rolling Upgrade involves complex orchestration as well as side-by-side installation. It is too complex for a manual procedure, and is therefore supported only as an Ambari feature. If you wish to perform a Rolling Upgrade, refer to the Ambari Install instructions to install Ambari, then follow the Ambari Rolling Upgrade instructions, see Ambari Upgrade Guide.

The HDP packages for a complete installation of HDP 2.6.0 occupies about 6.5 GB of disk space.

Warning

Until the upgrade is finalized, no HDFS data is deleted from the cluster. Be sure to review your capacity and ensure that you have extra space available during the upgrade window.

The following provides an overview of steps for upgrading to the latest release of HDP 2.6.x from HDP 2.5:

1. Get ready to upgrade
2. Upgrade HDP 2.5 Components and stop all services
3. Use hdp-select to symlink the HDP 2.6.0 components into "current," in place of the former HDP 2.5 components
4. Configure and Start Apache ZooKeeper
5. Configure and Start Hadoop
6. Start HDFS
7. Configure and start YARN/MapReduce
8. Configure and Start Apache HBase
9. Configure and Start Apache Phoenix
10. Configure and Start Apache Accumulo

11. Configure and Start Apache Tez
12. Configure and Start Apache Hive and Apache HCatalog
13. Configure and Start Apache Oozie
14. Configure and Start Apache WebHCat (Templeton)
15. Configure and Start Apache Pig
16. Configure and Start Apache Sqoop
17. Configure and Start Apache Flume
18. Configure and Start Apache Mahout
19. Configure and Start Apache Hue
20. Configure and Start Apache Knox
21. Configure and Start Apache Falcon
22. Configure and Start Apache Storm
23. Configure and Start Apache Ranger
24. Configure and Start Apache Spark
25. Upgrade Apache Slider
26. Upgrade Apache Kafka
27. Configure and Upgrade Apache Zeppelin
28. Finalize the Upgrade
29. Install new HDP 2.6 services

1.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.5 to HDP 2.6 versions and adding the new HDP 2.6 services. These instructions change your configurations.

Note

You must use **kinit** before running the commands as any particular user.

Hardware recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. The HDP packages for a complete installation of HDP 2.6.0 consumes about 6.5 GB of disk space.

The first step is to ensure you keep a backup copy of your HDP 2.5 configurations.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Back up the HDP directories for any hadoop components you have installed.

   The following is a list of all HDP directories:

   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hive-hcatalog/conf`
   - `/etc/hive-webhcat/conf`
   - `/etc/accumulo/conf`
   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/hue/conf`
   - `/etc/knox/conf`
   - `/etc/zookeeper/conf`
   - `/etc/tez/conf`
   - `/etc/storm/conf`
   - `/etc/falcon/conf`
   - /etc/slider/conf/
   - `/etc/ranger/admin/conf, /etc/ranger/usersync/conf` (If Ranger is installed, also take a backup of install.properties for all the plugins, ranger admin & ranger usersync.)
   - Optional - Back up your userlogs directories, `${mapred.local.dir}/userlogs`.

2. Oozie runs a periodic purge on the shared library directory. The purge can delete libraries that are needed by jobs that started before the upgrade began and

that finish after the upgrade. 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 prior to performing the upgrade:

```
<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>
```

3. Stop all long-running applications deployed using Slider:

   ```
   su - yarn "usr/hdp/current/slider-client/bin/slider list"
   ```

   For all applications returned in previous command, run `su - yarn "/usr/hdp/current/slider-client/bin/slider stop <app_name>"`

4. Stop all services (including MapReduce) except HDFS, ZooKeeper, and Ranger, and client applications deployed on HDFS.

   See Stopping HDP Services for more information.

Component	Command
Accumulo	`/usr/hdp/current/accumulo-client/bin$ / usr/hdp/current/accumulo-client/bin/stop-all.sh`
Knox	`cd $GATEWAY_HOME su - knox -c "bin/gateway.sh stop"`
Falcon	`su - falcon "/usr/hdp/current/falcon-server/bin/falcon-stop"`
Oozie	`su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-stop.sh`
WebHCat	`su - webhcat -c "/usr/hdp/hive-webhcat/sbin/webhcat_server.sh stop"`

Hive	Run this command on the Hive Metastore and Hive Server2 host machine:
	
	`ps aux | awk '{print $1,$2}' | grep hive | awk '{print $2}' | xargs kill >/dev/null 2>&1`
	
	Or you can use the following:
	
	`Killall -u hive -s 15 java`

Component	Command
HBase RegionServers	`su - hbase -c "/usr/hdp/current/hbase-regionserver/bin/hbase-daemon.sh --config /etc/hbase/conf stop regionserver"`
HBase Master host machine	`su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh --config /etc/hbase/conf stop master"`

YARN & Mapred Histro	Run this command on all NodeManagers:
	`su - yarn -c "export /usr/hdp/current/hadoop-client/libexec && /usr/hdp/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop nodemanager"`
	Run this command on the History Server host machine:
	`su - mapred -c "export /usr/hdp/current/hadoop-client/libexec && && /usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-jobhistory-daemon.sh --config /etc/hadoop/conf stop historyserver"`
	Run this command on the ResourceManager host machine(s):
	`su - yarn -c "export /usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop resourcemanager"`
	Run this command on the ResourceManager host machine:
	`su -yarn -c "export /usr/hdp/current/hadoop-client/libex && /usr/hdp/current/hadoop-yarn-timelineserver/sbin/yar-daemon.sh --config /etc/hadoop/conf stop timelineserver"`
	Run this command on the YARN Timeline Server node:
	`su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop timelineserver"`

Storm	`storm kill topology-name` `sudo service supervisord stop`
Spark (History server)	`su - spark -c "/usr/hdp/current/spark-client/sbin/stop-history-server.sh"`

5. If you have the Hive component installed, back up the Hive Metastore database.

   The following instructions are provided for your convenience. For the latest backup instructions, see your database documentation.

Table 1.1. Hive Metastore Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sqlsbr	mysql $dbname < $inputfilename.sqlsbr

Database Type	Backup	Restore
	For example: mysqldump hive > /tmp/mydir/backup_hive.sql	For example: mysql hive < /tmp/mydir/backup_hive.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql sbr For example: sudo -u postgres pg_dump hive > /tmp/mydir/backup_hive.sql	sudo -u $username psql $databasename < $inputfilename.sqlsbr For example: sudo -u postgres psql hive < /tmp/mydir/backup_hive.sql
Oracle	Export the database: exp username/password@database full=yes file=output_file.dmp	Import the database: imp username/password@database file=input_file.dmp

6. If you have the Oozie component installed, back up the Oozie metastore database.

   These instructions are provided for your convenience. Check your database documentation for the latest backup instructions.

Table 1.2. Oozie Metastore Database Backup and Restore

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

7. **Optional**: Back up the Hue database.

   The following instructions are provided for your convenience. For the latest backup instructions, please see your database documentation. For database types that are not listed below, follow your vendor-specific instructions.

Table 1.3. Hue Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sqlsbr For example:	mysql $dbname < $inputfilename.sqlsbr For example:

Database Type	Backup	Restore
	mysqldump hue > /tmp/mydir/backup_hue.sql	mysql hue < /tmp/mydir/backup_hue.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql sbr For example: sudo -u postgres pg_dump hue > /tmp/mydir/backup_hue.sql	sudo -u $username psql $databasename < $inputfilename.sqlsbr For example: sudo -u postgres psql hue < /tmp/mydir/backup_hue.sql
Oracle	Connect to the Oracle database using sqlplus. Export the database. For example: exp username/password@database full=yes file=output_file.dmp mysql $dbname < $inputfilename.sqlsbr	Import the database: For example: imp username/password@database file=input_file.dmp
SQLite	/etc/init.d/hue stop su $HUE_USER mkdir ~/hue_backup sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak /etc/init.d/hue start	/etc/init.d/hue stop cd /var/lib/hue mv desktop.db desktop.db.old sqlite3 desktop.db < ~/hue_backup/desktop.bak /etc/init.d/hue start

8. Back up the Knox data/security directory.

```
cp -RL /etc/knox/data/security ~/knox_backup
```

9. Save the namespace by executing the following commands:

```
su - hdfs

hdfs dfsadmin -safemode enter

hdfs dfsadmin -saveNamespace
```

Note

In secure mode, you must have Kerberos credentials for the hdfs user.

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

```
su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-old-fsck-1.log"
```

Note

In secure mode, you must have Kerberos credentials for the hdfs user.

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

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

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

Important

Make sure the namenode is started.

```
su - hdfs -c "hdfs dfs -ls -R / > dfs-old-lsr-1.log"
```

Note

In secure mode you must have Kerberos credentials for the hdfs user.

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

```
su - hdfs dfsadmin -c "-report > dfs-old-report-1.log"
```

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

```
su - hdfs -c "hdfs dfsadmin -report > dfs-old-report-l.log"
```

d. **Optional:** You can copy all or unrecoverable only data storelibext-customer directory in HDFS to a local file system or to a backup instance of HDFS.

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

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

```
su - hdfs -c "hdfs dfsadmin -finalizeUpgrade"
```

Note

In secure mode, you must have Kerberos credentials for the hdfs user.

13. Stop remaining services (HDFS, ZooKeeper, and Ranger).

See Stopping HDP Services for more information.

Component	Command
HDFS	On all DataNotes: If you are running secure cluster, run following command as root: `/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode` Else:

Component	Command

	`su - hdfs -c "usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode"`
	
	If you are not running a highly available HDFS cluster, stop the Secondary NameNode by executing this command on the Secondary NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop secondarynamenode"`
	
	On the NameNode host machine(s)
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop namenode"`
	
	If you are running NameNode HA, stop the ZooKeeper Failover Controllers (ZKFC) by executing this command on the NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-clinent/sbin/hadoop-deamon.sh --config /etc/hadoop/conf stop zkfc"`
	
	If you are running NameNode HA, stop the JournalNodes by executing these command on the JournalNode host machines:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh ==config /etc/hadoop/conf stop journalnode"`

ZooKeeper Host machines	`su - zookeeper "/usr/hdp/current/zookeeper-server/bin/zookeeper-server stop"`

Ranger (XA Secure)	`service ranger-admin stop`
	
	`service ranger-usersync stop`

14. Back up your NameNode metadata.

Note

It's recommended to take a backup of the full `/hadoop.hdfs/namenode` path.

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

The NameNode metadata is stored in a directory specified in the hdfs-site.xml configuration file under the configuration value "dfs.namenode.dir".

For example, if the configuration value is:

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/hadoop/hdfs/namenode</value>
</property>
```

Then, the NameNode metadata files are all housed inside the directory `/hadooop.hdfs/namenode`.

b. Store the layoutVersion of the namenode.

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

15. Verify that edit logs in `${dfs.namenode.name.dir}/current/edits*` are empty.

a. Run: `hdfs oev -i ${dfs.namenode.name.dir}/current/edits_inprogress_* -o edits.out`

b. Verify the edits.out file. It should only have OP_START_LOG_SEGMENT transaction. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<EDITS>
<EDITS_VERSION>-56</EDITS_VERSION>
<RECORD>
<OPCODE>OP_START_LOG_SEGMENT</OPCODE>
<DATA>
<TXID>5749</TXID>
</DATA>
</RECORD>
```

c. If edits.out has transactions other than OP_START_LOG_SEGMENT, run the following steps and then verify edit logs are empty.

- Start the existing version NameNode.
- Ensure there is a new FS image file.
- Shut the NameNode down:

```
hdfs dfsadmin – saveNamespace
```

16. Rename or delete any paths that are reserved in the new version of HDFS.

When upgrading to a new version of HDFS, it is necessary to rename or delete any paths that are reserved in the new version of HDFS. If the NameNode encounters a reserved path during upgrade, it prints an error such as the following:

```
/.reserved is a reserved path and .snapshot is a reserved path
component in this version of HDFS.

Please rollback and delete or rename this path, or upgrade with the
-renameReserved key-value pairs option to automatically rename these
paths during upgrade.
```

Specifying `-upgrade -renameReserved` optional key-value pairs causes the NameNode to automatically rename any reserved paths found during startup.

For example, to rename all paths named `.snapshot` to `.my-snapshot` and change paths named `.reserved` to `.my-reserved`, specify `-upgrade -renameReserved .snapshot=.my-snapshot,.reserved=.my-reserved`.

If no key-value pairs are specified with `-renameReserved`, the NameNode then suffixes reserved paths with:

`.<LAYOUT-VERSION>.UPGRADE_RENAMED`

For example: `.snapshot.-51.UPGRADE_RENAMED`.

Note

We recommend that you perform a `-saveNamespace` before renaming paths (running `-saveNamespace` appears in a previous step in this procedure). This is because a data inconsistency can result if an edit log operation refers to the destination of an automatically renamed file.

Also note that running `-renameReserved` renames all applicable existing files in the cluster. This may impact cluster applications.

17. If you are on JDK 1.6, upgrade the JDK on all nodes to JDK 1.7 or JDK 1.8 before upgrading HDP.

1.2. Upgrade HDP 2.5 Components

Important

See the Release Notes for the HDP 2.6.1.0 repo information.

The upgrade process to HDP 2.6.0 involves the following steps.

Select your OS:

RHEL/CentOS/Oracle 6

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.5 components. This command uninstalls the HDP 2.5 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons "bigtop""
   ```

3. Validate that all HDP 2.5 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.5
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/yum.repos.d/hdp.repo
   ```

5. Install the HDP 2.6.0 repo:

   - Download the hdp.repo file:

     ```
     ?????wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/
     updates/2.6.1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

- Confirm the HDP repository is configured.

  ```
  yum repolist
  ```

  You should see something like this. Verify that you have the HDP-2.6.0 directory:

  ```
  Loaded plugins: fastestmirror, security
  Loading mirror speeds from cached hostfile
  * base: mirrors.cat.pdx.edu
  * extras: linux.mirrors.es.net
  * updates: mirrors.usc.edu
  repo id repo namestatus
  HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
  ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

   ```
   yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "hive-webhcat" "hive-hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez"
    "storm" "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark"
    "slider" "hdp_mon_nagios_addons"
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

   ```
   yum list installed | grep HDP-<old.stack.version.number>
   ```

   No component file names should appear in the returned list.

RHEL/CentOS/Oracle 5 (Deprecated)

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.5 components. This command uninstalls the HDP 2.5 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.5 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.5
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/yum.repos.d/hdp.repo
   ```

5. Install the HDP 2.6.0 repo:

- Download the hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.6.
1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
```

- Confirm the HDP repository is configured.

```
yum repolist
```

You should see something like this. Verify that you have the HDP-2.6.0 directory:

```
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
* base: mirrors.cat.pdx.edu
* extras: linux.mirrors.es.net
* updates: mirrors.usc.edu
repo id repo namestatus
HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

```
yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie" "collectd"
 "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez" "storm"
 "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
 "hdp_mon_nagios_addons"
```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

```
yum list installed | grep HDP-<old.stack.version.number>
```

No component file names should appear in the returned list.

SLES 11 SP1

1. On all hosts, clean the yum repository.

```
zypper clean -all
```

2. Remove your old HDP 2.5 components. This command uninstalls the HDP 2.5 components. It leaves the user data, and metadata, but removes your configurations:

```
zypper rm "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
 "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
 "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
 "spark*" "slider*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.5 component binaries are uninstalled:

```
yum list installed | grep @HDP2.5
```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/sles11sp1/2.x/updates/2.6.
   1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez"
    "storm" "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark"
    "slider*" "hdp_mon_nagios_addons"
   ```

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

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-client
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

SLES 11 SP3/SP4, SLES 12

1. On all hosts, clean the zypper repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 2.5 components.

   ```
   zypper rm "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*" "gccxml*"
    "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*"
    "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue" "hue-common"
    "hue-shell" "knox*" "ranger*" "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.5 component binaries are uninstalled:

   ```
   zypper search --installed-only --repo HDP-2.5.0.0
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
   updates/2.6.1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "oozie" "collectd" "gccxml"
    "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez" "storm" "falcon"
    "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "spark-python"
    "hdp_mon_nagios_addons "slider*" "hive-webcat" "hive-hcatalog""
   ```

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-clientt
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

Ubuntu 12

1. On all hosts, clean the apt-get repository.

   ```
   apt-get clean -&-all
   ```

2. Remove your old HDP 2.5 components. This command uninstalls the HDP 2.5 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   apt-get remove "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
    "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez.
   *" "storm*" "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue.*"
    "knox*" "spark*" "slider*" "hdp_mon_nagios_addons" --purge
   ```

3. Validate that all HDP 2.5 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.5
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/apt/sources.list.d/hdp.list
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/
   updates/2.6.1.0.0/hdp.list - O /etc/apt/sources.list.d/hdp.list
   ```

6. Run an update:

   ```
   apt-get update
   ```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

   ```
   apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

Debian

1. On all hosts, clean the apt-get repository.

   ```
   apt-get clean -&-all
   ```

2. Remove your old HDP 2.5 components. This command uninstalls the HDP 2.5 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   apt-get remove "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
    "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*"
    "storm*" "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue*" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.5 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.5
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/apt/sources.list.d/hdp.list
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/
   debian<version>/2.x/updates/2.6.1.0.0/hdp.list - O /etc/apt/
   sources.list.d/hdp.list
   ```

6. Run an update:

   ```
   apt-get update
   ```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.5 components:

   ```
   apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

1.3. Symlink Directories with hdp-select

Warning

HDP 2.6.0 installs hdp-select automatically with the installation or upgrade of the first HDP component. If you have not already upgraded ZooKeeper, hdp-select has not been installed.

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.

1. Before you run hdp-select, remove one link:

   ```
   rm /usr/bin/oozie
   ```

2. Run hdp-select set all on your NameNode and all your DataNodes:

   ```
   hdp-select set all 2.6.1.0.0-<$version>
   ```

   For example:

   ```
   /usr/bin/hdp-select set all 2.6.1.0.0-2800
   ```

1.4. Configure and Start Apache ZooKeeper

Tip

If you are running a highly available HDFS cluster, configure and restart Apache ZooKeeper **before** you upgrade HDFS. This best practice lets the upgraded ZKFC work with your primary NameNode and your Standby NameNode.

1. Replace your configuration after upgrading on all the ZooKeeper nodes. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.

2. Start ZooKeeper.

   On all ZooKeeper server host machines, run the following command to start ZooKeeper and the ZKFC:

   ```
   su - zookeeper -c "/usr/hdp/current/zookeeper-server/bin/zookeeper-server
    start"
   ```

1.5. Configure Hadoop

RHEL/CentOS/Oracle Linux

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed in HDP 2.5. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

SLES

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed since HDP 2.5. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

Ubuntu/Debian

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml

2. Paths have changed in HDP 2.6.0. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

1.6. Start Hadoop Core

Warning

Before you start HDFS on a highly available HDFS cluster, you must start the ZooKeeper service. If you do not start the ZKFC, there can be failures.

To start HDFS, run commands as the $HDFS_USER.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using

another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Replace your configuration after upgrading on all the HDFS nodes. Replace the HDFS template configuration in `/etc/hdfs/conf`.

2. If you are upgrading from a highly available HDFS cluster configuration, start all JournalNodes. On each JournalNode host, run the following commands:

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

   

   Important

   All JournalNodes must be running when performing the upgrade, rollback, or finalization operations. If any JournalNodes are down when running any such operation, the operation fails.

3. If you are running HDFS on a highly available namenode, you must first start the ZooKeeper service

   

   Note

   Perform this step only if you are on a highly available HDFS cluster.

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

4. Start the NameNode.

   Because the file system version has now changed you must start the NameNode manually.

   On the active NameNode host, run the following commands:

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

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

   

   Note

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

   To check if the Upgrade is in progress, check that the "\previous" directory has been created in the \NameNode and \JournalNode directories. The "\previous" directory contains a snapshot of the data before upgrade.

   In a highly available HDFS cluster configuration, this NameNode does not enter the standby state as usual. Rather, this NameNode immediately enters the active state, perform an upgrade of its local storage directories, and also perform an upgrade of the

shared edit log. At this point, the standby NameNode in the HA pair is still down. It is out of sync with the upgraded active NameNode.

To synchronize the active and standby NameNode, re-establishing HA, re-bootstrap the standby NameNode by running the NameNode with the '-bootstrapStandby' flag. Do NOT start this standby NameNode with the '-upgrade' flag.

```
su - hdfs -c "hdfs namenode -bootstrapStandby -force"
```

The bootstrapStandby command downloads the most recent fsimage from the active NameNode into the $dfs.name.dir directory of the standby NameNode. You can enter that directory to make sure the fsimage has been successfully downloaded. After verifying, start the ZKFailoverController, then start the standby NameNode. You can check the status of both NameNodes using the Web UI.

5. Verify that the NameNode is up and running:

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

6. If you do not have a highly available HDFS cluster configuration (non_HA namenode), start the Secondary NameNode.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster configuration.

   On the Secondary NameNode host machine, run the following commands:

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

7. Verify that the Secondary NameNode is up and running.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster environment.

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

8. Start DataNodes.

   On each of the DataNodes, enter the following command. Note: If you are working on a `non-secure` DataNode, use $HDFS_USER. For a secure DataNode, use root.

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

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

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

10. Verify that NameNode can go out of safe mode.

```
>su - hdfs -c "hdfs dfsadmin -safemode wait"
```

You should see the following result: `Safe mode is OFF`

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

1.7. Verify HDFS Filesystem Health

Analyze if the filesystem is healthy.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

If you have a secure server, you need Kerberos credentials for hdfs user access.

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

   ```
   su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-new-fsck-1.log"
   ```

   You should see feedback that the filesystem under path / is HEALTHY.

2. Run `hdfs` namespace and report.

   a. List directories.

   ```
   su - hdfs -c "hdfs dfs -ls -R / > dfs-new-lsr-1.log"
   ```

   b. Open the `dfs-new-lsr-l.log` and confirm that you can see the file and directory listing in the namespace.

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

   ```
   su - hdfs -c "hdfs dfsadmin -report > dfs-new-report-1.log"
   ```

   d. Open the `dfs-new-report` file and validate the admin report.

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

   The file names are listed below:

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

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

Note

You must do this comparison manually to catch all errors.

4. From the NameNode WebUI, determine if all DataNodes are up and running.

   ```
   http://<namenode>:<namenodeport
   ```

5. If you are on a highly available HDFS cluster, go to the StandbyNameNode web UI to see if all DataNodes are up and running:

   ```
   http://<standbynamenode>:<namenodeport>
   ```

6. If you are **not** on a highly available HDFS cluster, go to the SecondaryNameNode web UI to see if it the secondary node is up and running:

   ```
   http://<secondarynamenode>:<secondarynamenodeport>
   ```

7. Verify that read and write to hdfs works successfully.

   ```
   hdfs dfs -put [input file] [output file]

   hdfs dfs -cat [output file]
   ```

1.8. Configure YARN and MapReduce

After you upgrade Hadoop, complete the following steps to update your configs.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

In secure mode, you must have Kerberos credentials for the hdfs user.

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

   ```
   su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/
   mapreduce/"

   su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/
   hadoop/mapreduce.tar.gz /hdp/apps/2.6.1.0.0-<$version>/
   mapreduce/"

   su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"

   su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<
   $version>/mapreduce"
   ```

```
su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<
$version>/mapreduce/mapreduce.tar.gz"
```

2. Make sure that the following properties are in `/etc/hadoop/conf/mapred-site.xml`:

   - Make sure mapreduce.application.framework.path exists in mapred-site.xml:

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

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

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

Note

If you are planning to use Spark in yarn-client mode, make Spark work in yarn-client mode 2.6.1.0.0-<$version>.

3. Make sure the following property is in `/etc/hadoop/conf/yarn-site.xml`:

```
<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. On secure clusters only, add the following properties to `/etc/hadoop/conf/yarn-site.xml`:

```
 <property>
  <name>yarn.timeline-service.recovery.enabled:</name>
  <value>TRUE</value>
  </property>

  <property>
  <name>yarn.timeline-service.state-store.class: org.apache.hadoop.yarn.
server.timeline.recovery:</name>
  <value>LeveldbTimelineStateStore</value>
  </property>

  <property>
  <name>yarn.timeline-service.leveldb-state-store.path:</name>
  <value><the same as the default of "yarn.timeline-service-leveldb-
timeline-store.path</value>
  </property>
```

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

   - Create the container-executor.cfg file in /usr/hdp/2.6.1.0.0-<version>/hadoop-yarn/bin/container-executor.

   - Insert the following properties:

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

- yarn.nodemanager.linux-container-executor.group - 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 - Comma-separated list of users who can not run container-executor.
- min.user.id - Minimum value of user id. This prevents system users from running container-executor.
- allowed.system.users - Comma-separated list of allowed system users.

- 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 run it:

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

chmod 6050 /usr/hdp/${hdp.version}/hadoop-yarn-server-nodemanager/bin/
container-executor
```

1.9. Start YARN/MapReduce Services

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

Note

The `su` commands in this section use "yarn" to represent the YARN Service user and mapreduce to represent the MAPREDUCE Service user. If you are using another name for these Service users, you need to substitute your Service user name for "yarn" or "mapreduce" in each of the `su` commands.

1. Manually clear the ResourceManager state store.

```
su - yarn -c "yarn resourcemanager -format-state-store"
```

2. Start the ResourceManager on all your ResourceManager hosts.

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

ps -ef | grep -i resourcemanager
```

3. Start the TimelineServer on your TimelineServer host.

```
su - yarn -c "/usr/hdp/current/hadoop-yarn-tinelineserver/sbin/yarn-daemon.
sh start timelineserver"

ps -ef | grep -i timelineserver
```

4. Start the NodeManager on all your NodeManager hosts.

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

ps -ef | grep -i nodemanager
```

5. To start MapReduce, run the following commands:

```
su - mapreduce -c "/usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-
jobhistory-daemon.sh start historyserver"

ps -ef | grep -i jobhistoryserver
```

1.10. Run Hadoop Smoke Tests

To smoke test your Hadoop upgrade, you can run the following MapReduce job as a regular user.

The job uses MapReduce to write 100MB of data into HDFS with RandomWriter

```
hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.
jar
            randomwriter -Dtest.randomwrite.total_bytes=10000000 test-after-
upgrade.
```

You should see messages similar to:

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

MapReduce upgraded successfully. You can now upgrade your other components.

Basic Troubleshooting

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

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

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

Accessing error messages:

1. Access the ApplicationMaster WebUI to view the container logs.

2. At your console logs for MapReduce job, look for a line with this format:

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

3. Select the logs link under ApplicationMaster table. It redirects you to the container logs. Error messages display here.

1.11. Configure and Start Apache HBase

Note

The `su` commands in this section use "hbase" to represent the HBASE Service user. If you are using another name for your HBASE Service user, you need to substitute your HBASE Service user name for "hbase" in each of the `su` commands.

The hbase.bucketcache.percentage.in.combinedcache is removed in HDP 2.6.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.5 needs to be recalculated to attain identical memory allotments in HDP 2.6.0. The L1 LruBlockCache is whatever hfile.block.cache.size is set to and the L2 BucketCache is whatever hbase.bucketcache.size is set to.

1. Replace your configuration after upgrading. Replace the Apache HBase template configuration in `/etc/hbase/conf`.
2. Start services. From root, assuming that $HBASE_USER=hbase:

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

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

3. Check processes.

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

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

4. If bulk load support for backup/restore is required, follow these steps:

   a. Register `org.apache.hadoop.hbase.backup.BackupHFileCleaner` through `hbase.master.hfilecleaner.plugins`.

      `org.apache.hadoop.hbase.backup.BackupHFileCleaner` is responsible for keeping bulk loaded hfiles so that incremental backup can pick them up.

   b. Register `org.apache.hadoop.hbase.backup.BackupObserver` through `hbase.coprocessor.region.classes`.

      `org.apache.hadoop.hbase.backup.BackupObserver` is notified when bulk load completes and writes records into `hbase:backup table`.

1.12. Configure Apache Phoenix

To configure Phoenix, complete the following steps:

1. Add the following property to the `/etc/hbase/hbase-site.xml` file on all HBase nodes, the MasterServer, and all RegionServers to prevent deadlocks from occurring during maintenance on global indexes:

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

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

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

3. Ensure the client side hbase-site.xml matches the server side configuration.

4. **(Optional)** To use local indexing, set the following three properties. These properties ensure collocation of data table and local index regions:

Note

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

Add the following two properties, if they do not already exist, to the master side configurations:

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

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

Add the following, if it does not already exist, to the RegionServer side configurations:

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

5. If the folder specified in `hbase.tmp.dir` property on `hbase-site.xml` does not exist, create that directory with adequate permissions.

6. Set the following porperty in the hbase-site.xml file for all RegionServers, but not on the client side:

```
<property>
  <name>hbase.rpc.controllerfactory.class</name>
  <value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</
value>
```

```
</property>
```

7. Restart the HBase Master and RegionServers.

Configuring Phoenix to Run in a Secure Cluster

Perform the following additional steps to configure Phoenix to run in a secure Hadoop cluster:

1. To link the HBase configuration file with the Phoenix libraries:

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

2. To link the Hadoop configuration file with the Phoenix libraries:

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

Note

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, ignored java.io.IOException:
No FileSystem for scheme: hdfs
```

1.13. Configure and Start Apache Accumulo

Note

The `su` commands in this section use "accumulo" to represent the Accumulo Service user. If you are using another name for your Apache Accumulo Service user, you need to substitute your Accumulo Service user name for "accumulo" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/accumulo/conf` from the template to the conf directory in Accumulo hosts.

2. Start the services:

   ```
   su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
    `hostname` master"

   su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
    `hostname` tserver"

   su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
    `hostname` gc"
   ```

```
su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` tracer"

su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` monitor"
```

3. Check that the processes are running

   ```
   ps -ef | grep accumulo
   ```

   or visit http://<hostname>:50095 in your browser

1.14. Configure and Start Apache Tez

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

To upgrade Apache Tez:

1. Copy your previously backed-up copy of tez-site.xml into the /etc/tez/conf directory.
2. Upload the Tez tarball to HDFS.

   ```
   su - hdfs
   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:hadoop /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 <hdp_version> is the current HDP version, for example 2.6.1.0.0-2800.

3. Edit the tez.lib.uris property in the tez-site.xml file to point to /hdp/apps/ <hdp_version>/tez/tez.tar.gz

   ```
   ...
   <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, for example `2.6.1.0.0-2800`.

4. **Optional** Earlier releases of Tez did not have access control. In the current version of Tez, the default behavior restricts the ability to view the Tez history to only the owner of the job. To retain unrestricted access for non-secure clusters, set `tez.am.view-acls` set to "*".
5. Change the value of the tez.tez-ui.history-url.base property to the url for the upgraded Tez View. For information on setting up the Tez view, see Deploying the Tez View in the HDP Ambari Views Guide.

6. **Run Tez Smoke Test**

   To smoke test your Tez upgrade, you can run the following Tez Example job as a regular user.

   MRRSleep Tez Example job sleeps for a defined period of time in mapper and reducer

   ```
   hadoop jar /usr/hdp/current/tez-client/tez-tests-<version>.jar mrrsleep -m 1
    -r 1 -mt 100 -rt 100
   ```

   You should see messages similar to:

   ```
   …DAG: State: SUCCEEDED Progress: 100%
   DAG completed. FinalState=SUCCEEDED
   ```

   Tez upgraded successfully. You can now upgrade your other components.

1.15. Configure and Start Apache Hive and Apache HCatalog

Note

The su commands in this section use "hive" to represent the Hive Service user. If you are using another name for your Hive Service user, you need to substitute your Hive Service user name for "hive" in each of the su commands.

1. Prior to starting the upgrade process, set the following in your hive configuration file:

   ```
   datanucleus.autoCreateSchema=false
   ```

2. Copy the jdbc connector jar from OLD_HIVE_HOME/lib to CURRENT_HIVE_HOME/lib.

3. Upgrade the Apache Hive Metastore database schema. Restart the Hive Metastore database and run:

   ```
   su - hive -c "/usr/hdp/current/hive-metastore/bin/schematool -upgradeSchema
    -dbType" <$databaseType>
   ```

   The value for $databaseType can be derby, mysql, oracle, or postgres.

Note

If you are using Postgres 8 and Postgres 9, you should reset the Hive Metastore database owner to <HIVE_USER>:

```
psql -U <POSTGRES_USER> -c
ALTER DATABASE <HIVE-METASTORE-DB-NAME> OWNER TO <HIVE_USER>
```

Note

If you are using Oracle 11, you may see the following error message:

```
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
optimize.mapjoin.mapreduce does not exist
```

```
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
heapsize does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
server2.enable.impersonation does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
semantic.analyzer.factory.impl does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.auto.
convert.sortmerge.join.noconditionaltask does not exist
Metastore connection URL: jdbc:oracle:thin:@//ip-172-31-42-1.ec2.
internal:1521/XE
Metastore Connection Driver : oracle.jdbc.driver.OracleDriver
Metastore connection User: hiveuser
Starting upgrade metastore schema from version 0.13.0 to 0.14.0
Upgrade script upgrade-0.13.0-to-0.14.0.oracle.sql
Error: ORA-00955: name is already used by an existing object
 (state=42000,code=955)
Warning in pre-upgrade script pre-0-upgrade-0.13.0-to-0.14.0.
oracle.sql: Schema script failed, errorcode 2
Completed upgrade-0.13.0-to-0.14.0.oracle.sql
schemaTool completed
```

You can safely ignore this message. The error is in the pre-upgrade script and can be ignored; the schematool succeeded.

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

   a. Edit the following properties in the hive-site.xml file:

```
<property>
 <name>fs.file.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.file.impl.disable.cache</
description>
</property>

<property>
 <name>fs.hdfs.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.hdfs.impl.disable.cache
 </description>
<property>
```

   b. **Optional:** To enable the Hive buildin authorization mode, make the following changes. If you want to use the advanced authorization provided by Ranger, refer to the Ranger instructions.

      Set the following Hive authorization parameters in the hive-site.xml file:

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

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

```
</property>

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdConfOnlyAuthorizeFactory</value>
</property>
```

Also set hive.users.in.admin.role to the list of comma-separated users who need to be added to admin role. A user who belongs to the admin role needs to run the "set role" command before getting the privileges of the admin role, as this role is not in the current roles by default.

Set the following in the hiveserver2-site.xml file.

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

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

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdHiveAuthorizeFactory/value>
</property>
```

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

```
<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>
```

You can further fine-tune your configuration settings based on node hardware specifications, using the HDP utility script.

5. Start Hive Metastore.

   On the Hive Metastore host machine, run the following command:

   ```
   su - hive -c "nohup /usr/hdp/current/hive-metastore/bin/hive
   --service metastore -hiveconf hive.log.file=hivemetastore.log
   >/var/log/hive/hivemetastore.out 2>/var/log/hive/
   hivemetastoreerr.log &"
   ```

6. Start Hive Server2.

On the Hive Server2 host machine, run the following command:

```
su - hive

nohup /usr/hdp/current/hive-server2/bin/hiveserver2 -hiveconf
hive.metastore.uris=" " -hiveconf hive.log.file=hiveserver2.log
>/var/log/hive/hiveserver2.out 2> /var/log/hive/
hiveserver2err.log &
```

1.16. Configure and Start Apache Oozie

Note

The duration of the Oozie upgrade is dependent on the amount of job history stored in ooziedb. This history must be backed up and restored during the upgrade process. Best practice when planning for upgrade is to backup ooziedb from your production oozie server and restore it to a test or development oozie server. This can help you estimate the time that is be required to upgrade Oozie during your production upgrade.

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and "oozie" to represent the Oozie Service user. If you are using another name for your HDFS Service user or your Oozie Service user, you need to substitute your Service user names for "hdfs" or "oozie" in each of the `su` commands.

Upgrading Apache Oozie is a complex process. Although the instructions are straightforward, set aside a dedicated block of time to upgrade oozie clients and servers.

Perform the following preparation steps on each oozie server host:

1. You must restore oozie-site.xml from your backup to the conf directory on each oozie server and client.

2. Copy the JDBC jar to libext-customer:

   a. Create the /usr/hdp/current/oozie/libext-customer directory.

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

   mkdir libext-customer
   ```

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

   ```
   chmod -R 777 /usr/hdp/current/oozie-server/libext-customer
   ```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/current/hadoop-client/lib/hadoop*lzo*.jar /usr/hdp/
   current/oozie-server/libext-customer
   ```

```
cp /usr/share/HDP-oozie/ext.zip /usr/hdp/current/oozie-server/
libext-customer/
```

Also, copy Oozie db jar in libext-customer.

4. If Falcon was also installed and configured before upgrade in HDP 2.5.x, then after upgrade you might also need to do the following:

```
cp /usr/hdp/current/falcon-server/oozie/ext/falcon-oozie-el-
extension-"jar /usr/hdp/current/oozie-server/libext-customer
```

5. Extract share-lib.

```
/usr/hdp/current/oozie/bin/oozie-setup.sh sharelib create -fs
hdfs://<namenode>:8020
```

To verify that the sharelibs extracted correctly, run the following command:

```
oozie admin -oozie http://<oozie server host address>:11000/
oozie -shareliblist
```

There should be:

- Available ShareLib
- oozie
- hive
- distcp
- hcatalog
- sqoop
- mapreduce-streaming
- pig

Change the ownershop and permissions of the oozie directory:

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

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

Add the two zip files from spark-client phone to spark sharelib:

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

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

6. If a previous version of Oozie was created using auto schema creation, run the following SQL query:

```
insert into oozie_sys (name, data) values ('db.version', '2.6');
```

7. As the Oozie user (not root), run the upgrade.

   ```
   su - oozie -c "/usr/hdp/current/oozie-server/bin/ooziedb.sh
   upgrade -run"
   ```

8. As root, prepare the Oozie WAR file.

   ```
   chown oozie:oozie /usr/hdp/current/oozie-server/oozie-server/
   conf/server.xml
   ```

   ```
   su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-setup.sh
   prepare-war -d /usr/hdp/current/oozie-server/libext-customer"
   ```

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

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

9. Make sure that following property is added in oozie-log4j.properties:

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

   where ${oozie.instance.id} is determined by oozie, automatically.

10. If you have custom Oozie actions, you must define them in 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>
    <property>
         <name>oozie.service.SparkConfigurationService.spark.configuration</
    name>
         <value>*=/etc/spark/conf/</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>
    ```

11. Configure HTTPS for the Oozie server.

    a. Create a self signed certificate or get certificate from a trusted CA for the Oozie Server

    b. Import the certificate to the client JDK trust store on all client nodes.

c. In the Ambari Oozie configuration, set the following environment variables in `oozie-env.sh`, adding them if it does not exist:

```
export OOZIE_HTTPS_PORT=11443
export OOZIE_HTTPS_KEYSTORE_FILE=/home/oozie/.keystore
export OOZIE_HTTPS_KEYSTORE_PASS=password
```

d. Change `OOZIE_HTTP_PORT={{oozie_server_port}}` to `OOZIE_HTTP_PORT=11000`.

e. Set the `oozie.base.url` to the HTTPS address.

f. Save the configuration, and restart the Oozie components.

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

13. Check processes.

```
ps -ef | grep -i oozie
```

14. When needed, the `/usr/hdp/current/oozie-server/bin/oozied.sh` script with the *`stop`* parameter stops the Oozie server.

1.17. Configure and Start Apache WebHCat

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and webhcat to represent the WebHCat Service user. If you are using another name for these Service users, you need to substitute your Service user name for "hdfs" or "webhcat" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/webhcat/conf` from the template to the conf directory in webhcat hosts.

2. Modify the Apache WebHCat configuration files.

   a. Upload Pig, Hive and Sqoop tarballs to HDFS as the $HDFS_User (in this example, hdfs):

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/pig/pig.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/hive/hive.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/hive/"
```

```
su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/sqoop/sqoop.
tar.gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/pig"

su - hdfs - "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/pig/
pig.tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/hive"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/hive/
hive.tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/
sqoop/sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"
```

b. Update the following properties in the webhcat-site.xml configuration file, as their values have changed:

```
<property>
 <name>templeton.pig.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/pig/pig.tar.gz</value>
</property>

<property>
 <name>templeton.hive.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/hive/hive.tar.gz</value>
</property>

<property>
 <name>templeton.streaming.jar</name>
 <value>hdfs:///hdp/apps/${hdp.version}/mapreduce/
   hadoop-streaming.jar</value>
 <description>The hdfs path to the Hadoop streaming jar file.</
description>
</property>

<property>
 <name>templeton.sqoop.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/sqoop/sqoop.tar.gz</value>
 <description>The path to the Sqoop archive.</description>
</property>

<property>
 <name>templeton.sqoop.path</name>
 <value>sqoop.tar.gz/sqoop/bin/sqoop</value>
 <description>The path to the Sqoop executable.</description>
</property>

<property>
 <name>templeton.sqoop.home</name>
 <value>sqoop.tar.gz/sqoop</value>
 <description>The path to the Sqoop home in the exploded archive.
   </description>
</property>
```

Note

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

c. Add the following property if it is not present in webhcat-sitemxml:

```
<property>
 <name>templeton.libjars</name>
 <value>/usr/hdp/current/zookeeper-client/zookeeper.jar,/usr/hdp/current/
hive-client/lib/hive-common.jar</value>
 <description>Jars to add the classpath.</description>
</property>
```

d. Remove the following obsolete properties from webhcat-site.xml:

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

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

e. Add new proxy users, if needed. In core-site.xml, make sure the following properties are also set to allow WebHCat to impersonate your additional HDP 2.6.0 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 by 'hcat'.

`hadoop.proxyuser.hcat.hosts`

A comma-separated list of the hosts which are allowed to submit requests by 'hcat'.

3. Start WebHCat:

```
su - webhcat -c "/usr/hdp/current/hive-webhcat/sbin/
webhcat_server.sh start"
```

4. Smoke test WebHCat.

a. If you have a non-secure cluster, on the WebHCat host machine, run the following command to check the status of WebHCat server:

```
curl http://$WEBHCAT_HOST_MACHINE:50111/templeton/v1/status
```

You should see the following return status:

```
"status":"ok","version":"v1"
```

b. If you are using a Kerberos secure cluster, run the following command:

```
curl --negotiate -u: http://$WEBHCAT_HOST_MACHINE:50111/
templeton/v1/status
```

You should see the following return status

```
{"status":"ok","version":"v1"}[machine@acme]$
```

1.18. Configure Apache Pig

Replace your configuration after upgrading. Copy /etc/pig/conf from the template to the conf directory in pig hosts.

1.19. Configure and Start Apache Sqoop

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Replace your configuration after upgrading. Copy /etc/sqoop/conf from the template to the conf directory in sqoop hosts.
2. As the HDFS Service user, upload the Apache Sqoop tarball to HDFS.

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.6.1.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/sqoop/sqoop.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chmod 444 /hdp/apps/2.6.1.0.0-<$version>/sqoop/
sqoop.tar.gz"
```

3. If you are using the MySQL database as a source or target, then the MySQL connector jar must be updated to 5.1.29 or later.

   Refer to the MySQL web site for information on updating the MySQL connector jar.

1.20. Configure, Start, and Validate Apache Flume

1. Replace your configuration after upgrading. Copy /etc/flume/conf from the template to the conf directory in Flume hosts.
2. By default, Apache Flume does not start running immediately upon installation.
   To validate your Flume upgrade, replace your default conf/flume.conf with the provided flume.conf file, restart Flume, and see if the data is flowing by examining the destination.

   Use this flume.conf file:

```
#1. Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

#2.Describe/configure the source
a1.sources.r1.type = seq

#3. Describe the sink
a1.sinks.k1.type = file_roll
a1.sinks.k1.channel = c1
a1.sinks.k1.sink.directory = /tmp/flume

#4. Use a channel which buffers events in memory
a1.channels.c1.type = memory

#5. Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

3. After starting Flume, check /tmp/flume to see if there are any files there. The files should contain simple sequential numbers.
4. After validating, stop Flume and revert changes to flume.conf to prevent your disk from filling up.

1.21. Configure, Start, and Validate Apache Mahout

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

Replace your configuration after upgrading. Copy /etc/mahout/conf from the template to the conf directory in mahout hosts.

To validate Apache Mahout:

1. Create a test user:

```
su - hdfs -c "dfs -put /tmp/sample-test.txt /user/testuser"
```

2. Set up mahout to convert the plain text file sample-test.txt into a sequence file that is in the output directory mahouttest.

```
mahout seqdirectory --input /user/testuser/sample-test.txt --output /user/
 testuser/mahouttest --charset utf-8
```

1.22. Configure and Start Hue

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

For HDP 2.6.0, use the Hue version shipped with HDP 2.6.0. If you have a previous version of Hue, use the following steps to upgrade Hue.

1. Migrate hue.ini setting from your old hue.ini configuration file to new hue.ini configuration file.
2. If you are using the embedded SQLite database, remember to restore your database after upgrade.

   To restore the database from a backup, make sure the destination database is empty before copying (if necessary, rename or remove the current destination database), then copy your backup to the destination database. For example:

```
su - hue
cd /var/lib/hue
mv desktop.db desktop.db.old
sqlite3 desktop.db < ~/hue_backup/desktop.bak
exit
```

3. Synchronize Database

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

4. Start Hue. As a root user, run the following command on the Hue Server:

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

1.23. Configure and Start Apache Knox

When working with the Apache Knox Gateway in your Hadoop cluster, it is important you have the latest version of Knox installed so you can take advantage of new features and enhancements, in addition to ensuring your instance of Knox is in sync with other Hadoop components (e.g. Ranger, Spark, Hive, Hue, etc.) for stability and performance. For example, if you need to upgrade your Hadoop cluster from 2.5 to 2.6, you should

also make sure that your individual Hadoop components are also upgraded to the latest version.

HDP enables you to perform a rolling upgrade in 2.2.x and onward. A rolling upgrade means that you can upgrade a component, or the entire Hadoop stack, without losing service, and your users can continue to use the cluster and run jobs with no application or server downtime. The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL.

The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL. Once the upgrade process is completed, you are up and running with the latest version of Knox on each server you have designated as a Knox server.

Note

In this document, whenever you see a {} with a value inside, this denotes a value you must define.

1.23.1. Upgrade the Knox Gateway

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Knox manually to the latest version. Because “rolling upgrades” are now supported in HDP 2.6.0, it is not important which version of Knox you are currently running, only that you have an instance of the Knox Gateway running.

Note

If you have not already installed Knox, refer to the "Install the Knox RPMs on the Knox Server" section of the *Non-Ambari Cluster Installation Guide* for instructions on how to install and configure the Knox Gateway.

Before upgrading the Knox Gateway, there are a several steps you must follow to ensure your configuration files, settings, and topology files can be copied to the new Knox Gateway instance when the upgrade is complete, which are described below.

1. Back up your existing `conf` directory if you have not already done so.
2. Stop each Knox server if you have not already done so.

   ```
   su -l knox /usr/hdp/{the current Knox version}/knox/bin/gateway.sh stop
   ```

3. Select the HDP server version you are upgrading to after you have stopped each Knox server if you have not already done so.

   ```
   hdp-select set knox-server {the HDP server version}
   ```

4. 

   Note

   The `su` commands in this section use "knox" to represent the Knox Service user. If you are using another name for your Knox Service user, you need

to substitute your Knox Service user name for "knox" in each of the `su` commands.

For HDP 2.6.0, the default paths for Knox change. Upgrade Knox in order to update these paths.

a. Restore the backed up security directory. This places the master secret and keystores back in place for the new deployment.

b. Start the Gateway:

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

c. Unzip your previously saved configuration directory (the `conf` directory you backed up in step 1) into the new */var/log/knox/gateway.conf* directory to import these files.

d. Restart the Knox server to complete the upgrade.

```
su -l knox /usr/hdp/{the new HDP server version}/knox/bin/gateway.sh
 start
```

1.23.2. Verify the Knox Upgrade

To verify the upgrade was successful, follow the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful.

2. Verify you have cluster access using the `LISTSTATUS` WebHDFS API call.

```
curl -ivk -u {user}:{password} https://{knox host}:8443 /gateway/webhdfs/v1/
tmp?op=LISTSTATUS
```

3. Verify the Knox version using the Knox Admin service and Version API.

```
curl -ivk -u {adminuser}:{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

Note

The Admin API requires you to be a member of an Admin group, as specified in the `admin.xml` authorization provider.

When you have verified the Knox upgrade was successful, you can begin using Knox. If, however, the upgrade was unsuccessful, you need to downgrade the Knox Gateway to the previous version. The steps to downgrade the Knox Gateway are described in the next section.

1.23.3. Downgrade the Knox Gateway to the Previous Version

If the Knox Gateway upgrade was unsuccessful, you need to downgrade Knox to the previous version to ensure you have a working Knox Gateway for your cluster. To downgrade Knox, follow the steps listed below.

1. For each server running Knox, stop the server.

```
su -l knox /usr/hdp/{current HDP server version}/knox/bin/gateway.sh stop
```

2. Select the HDP server version you want to use to downgrade your Knox Gateway.

```
hdp-select set knox-server {previous HDP server version}
```

3. Restart the server.

```
su -l knox /usr/hdp/{previous HDP server version}/knox/bin/gateway.sh start
```

1.23.4. Verify the Knox Downgrade Was Successful

When the restart is complete, verify you are running an older version of Knox by following the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful

2. Verify you have cluster access using the `LISTSTATUS`WebHDFS API call.

3. Check the Knox version using the Knox Admin service and Version API using the following command:

```
curl -ivk -u {adminuser}"{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

1.24. Configure and Validate Apache Falcon

Note

In HDP 2.6.0, if authorization is enabled (for example, in the properties file with `*.falcon.security.authorization.enabled=true`") then Access Control List (ACL) is mandated for all entities.

Upgrade Apache Falcon after you have upgraded HDFS, Hive, Oozie, and Pig. Stop Oozie jobs while running Falcon.

1. Replace your configuration after upgrading. Copy /etc/falcon/conf from the template to the conf directory in falcon hosts.

2. Check your Falcon entities. There should be no changes, but in some cases you may need to update your entities post-upgrade.

3. In HDP 2.6.0 for Falcon, TLS is enabled by default. When TLS is enabled, Falcon starts on https://<falcon_host>.15443/. You can disable TLS by adding the following line to the startup.properties file.

   `"*.falcon.enableTLS=false`

4. If Transport Layer Security (TLS) is disabled, check the client.properties file to make sure the property "falcon.uri" is set as follows:

```
falcon.uri=http://<falcon_host>:15000/
```

Troubleshooting:

When upgrading Falcon in HDP 2.5 or later, you might encounter the following error when starting the ActiveMQ server:

```
ERROR - [main:] ~ Failed to start ActiveMQ JMS Message Broker. Reason:
java.lang.NegativeArraySizeException (BrokerService:528)
```

If you encounter this error, follow these steps to delete the ActiveMQ history and then restart Falcon. If you want to retain the history, be sure to back up the ActiveMQ history prior to deleting it.

```
cd <ACTIVEMQ_DATA_DIR>
rm -rf ./localhost
cd /usr/hdp/current/falcon-server
su -l <FALCON_USER>
./bin/falcon-stop
./bin/falcon-start
```

1.25. Configure and Start Apache Storm

Note

The `su` commands in this section use "zookeeper" to represent the ZooKeeper Service user. If you are using another name for your ZooKeeper Service user, you need to substitute your ZooKeeper Service user name for "zookeeper" in each of the `su` commands.

Apache Storm is fairly independent of changes to the HDP cluster:

1. Deactivate all running topologies.
2. Delete all states under zookeeper:

   ```
   /usr/hdp/current/zookeeper-client/bin/zkCli.sh (optionally in
   secure environment specify -server zk.server:port)
   ```

3. `rmr /storm`
4. Delete all states under the storm-local directory:

   ```
   rm -rf <value of stormlocal.dir>
   ```

5. Stop Storm Services on the storm node.
6. Update the following configs in `storm.yaml`:

   - `storm.thrift.transport=org.apache.storm.security.auth.SimpleTransportPlugin`
   - `storm.messaging.transport=org.apache.storm.messaging.netty.Context`
   - `nimbus.topology=org.apache.storm.nimbus.DefaultTopologyValidator`

- `topology.spout.wait.strategy=org.apache.storm.spout.SleepSpoutWaitStrategy`
- `topology.kryo.factory=org.apache.storm.serialization.DefaultKryoFactory`
- `topology.tuple.serializer=org.apache.storm.serialization.types.ListDelegate`
- `nimbus.authorizer=org.apache.storm.security.suth.authorizer.SimpleACLAuthor` (applicable only in a secure cluster)
- `drpc.authorizer=org.apache.storm.security.auth.authorizer.DRPCSimpleACLAuth` (applicable only in a secure cluster)
- `ui.filter=org.apache.storm.secuity.auth.KerberosPrincipalToLocal` (applicable only in a secure cluster)

7. Stop ZooKeeper Services on the storm node.

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

8. Remove Storm and zookeeper from the storm node and install the HDP 2.6.0 version:
   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum erase storm

     yum erase zookeeper

     yum install storm

     yum install zookeeper
     ```

   - For **SLES**:

     ```
     zypper rm storm

     zypper rm zookeeper

     zypper install storm

     zypper install zookeeper
     ```

   - For **Ubuntu/Debian**:

     ```
     apt-get remove storm --purge

     apt-get remove zookeeper --purge

     apt-get install storm

     apt-get install zookeeper
     ```

9. Replace your configuration after upgrading. Copy `/etc/storm/conf` from the template to the conf directory .

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

11. Ensure ZooKeeper is running. On the storm node, run the following command:

```
su - zookeeper -c "source /etc/zookeeper/conf/zookeeper-env.sh; export
 ZOOCFGDIR=/etc/zookeeper/conf; /usr/hdp/current/zookeeper-server/bin/
zkServer.sh start >> $ZOO_LOG_DIR/zoo.out\"
```

where

- $ZOO_LOG_DIR is the directory where ZooKeeper server logs are stored. For example, /var/log/zookeeper.

12. Start nimbus, then supervisor/ui/drpc/logviewer:

```
/usr/hdp/current/storm-nimbus/bin/storm nimbus.
```

13. Start Storm, using a process controller, such as supervisor:

```
su - storm /usr/hdp/current/storm-supervisor/bin/storm
supervisor
```

You can use the same command syntax to start Storm using nimbus/ui and logviewer.

```
su - storm /usr/hdp/current/storm-supervisor/bin/storm nimbus

su - storm /usr/hdp/current/storm-supervisor/bin/storm ui

su - storm /usr/hdp/current/storm-supervisor/bin/storm logviewer

su - storm /usr/hdp/current/storm-supervisor/bin/storm drpc
```

1.26. Configure and Start Apache Ranger

This section describes how to upgrade the Ranger service.

1.26.1. Prerequisites

When using MySQL, the storage engine used for the Ranger admin policy store tables MUST support transactions. InnoDB is an example of engine that supports transactions. A storage engine that does not support transactions is not suitable as a policy store.

1.26.2. Preparing Your Cluster to Upgrade Ranger

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Apache Ranger manually to the latest version. This section describes the steps you need to follow to prepare your cluster for the Ranger upgrade.

1. Back up the following Ranger configuration directories:

   - Ranger Policy Administration Service

```
/etc/ranger/admin/conf
```

- Ranger UserSync

```
/etc/ranger/usersync/conf
```

- Ranger Plugins:
  - Kafka

```
/etc/kafka/conf
```

  - Atlas

```
 /etc/atlas/conf
```

  - YARN

```
/etc/hadoop/conf
```

  - Hadoop

```
/etc/hadoop/conf
```

  - Hive

```
/etc/hive/conf
```

  - HBase

```
/etc/hbase/conf
```

  - Knox

```
/etc/knox/conf
```

  - Storm

```
/etc/storm/conf
```

2. Backup the Ranger Policy. Backup the `install.properties` file. Make sure to take note of the following details in the `install.properties:` file:
   - db_host
   - db_name
   - db_user
   - db_password
   - policy manager configuration
   - LDAP directory configuration
   - LDAP settings
   - LDAP AD domain

- LDAP URL

```
mysqldump -u root -p root ranger > dest_dir/filename.sql
mysqldump -u root -p root ranger_audit > dest_dir/audit_filename.sql
```

1.26.3. Stop the Ranger Services

Now that you have prepared your cluster for the Ranger upgrade, you need to stop the Ranger Admin and Ranger UserSync services. To stop the Ranger services, perform the steps described below.

1. Stop the Ranger Policy Admin service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-admin stop
   ```

2. Stop the Ranger UserSync service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-usersync stop
   ```

3. Stop the applicable services using the Ranger plugin (HDFS, HBase, Hive, Knox, Storm).

   See Stopping HDP Services for more information.

1.26.4. Preparing the Cluster for Upgrade

Before you begin the upgrade process, you need to perform a series of steps to prepare the cluster for upgrade. These steps are described in the "*Getting Ready To Upgrade*" section of this guide, which you need to follow before continuing to upgrade Ranger. Some of these steps include:

- Backing up HDP directories
- Stopping all long-running applications and services.
- Backing up the Hive and Oozie metastore databases.
- Backing up Hue
- Backing up specific directories and configurations

1.26.5. Registering the HDP 2.6.0 Repo

After you have prepared your cluster for the upgrade, you need to register the HDP 2.6.0 repo. This requires you to perform the following steps:

1. (Optional) The Ranger components should already have installed in the at the beginning of the HDP upgrade process, but you can use the following commands to confirm that the Ranger packages have been installed:

   ```
   hdp-select status ranger-admin
   ```

```
hdp-select status ranger-usersync
```

If the packages have not been installed, you can use the install commands specific to your OS. For example, for RHEL/CentOS you would use the following commands to install the packages.

```
yum install ranger_2_5_*-admin
yum install ranger_2_5_*-usersync
```

2. Select the Ranger Admin and Ranger UserSync versions you want to use.

```
hdp-select set ranger-admin <HDP_server_version>
hdp-select set ranger-usersync <HDP_server_version>
```

3. Change ownership of `/etc/ranger/admin/conf/` and `/etc/ranger/usersync/conf/` to `ranger:ranger`:

```
chown -R ranger:ranger /etc/ranger/admin/conf/
```

4. Solr must be installed and configured before installing RangerAdmin or any of the Ranger component plugins.

   For information regarding installation and configuration of Solr, see Using Apache Solr for Ranger Audits.

5. Update the `install.properties` file to migrate the database credentials properties and POLICYMGR_EXTERNAL-URL property from HDP 2.5. to HDP 2.6.0.

Table 1.4. Ranger_Admin install.properties names and values

Property Name	Property Value
DB_FLAVOR	MySql(|ORACLE|POSTGRES|MSSQL|SQLA)
db_root_user	root
db_root_password	Password of db (eg: vagrant)
db_host	Hostname : where your db does exist
policymgr_external_url	http://<hostname>:6080
policymgr_http_enabled	true
authentication_method	UNIX(LDAP|ACTIVE_DIRECTORY|UNIX|NONE)
audit_store	solr
audit_solr_urls	http://<solr_host>:6083/solr/ranger_audits

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

6. If you are using an SSL-enabled, MySQL database, add the following properties to `install.properties`:

```
db_ssl_enabled=false
db_ssl_required=false
```

```
db_ssl_verifyServerCertificate=false
javax_net_ssl_keyStore=
javax_net_ssl_keyStorePassword=
javax_net_ssl_trustStore=
javax_net_ssl_trustStorePassword=
```

For additional information, see SSL Enabled MySQL.

7. If Ranger Admin is SSL-enabled, add the follow Ranger Admin SSL properties to the `install.properties` file. These properties secure the Ranger SSL password in the `jceks` file.

```
policymgr_https_keystore_file=<SSL keystore file path used to configure
 Ranger in SSL>
policymgr_https_keystore_keyalias=rangeradmin
policymgr_https_keystore_password=<SSL password used to create keystore>
```

8. The RANGER_PID_DIR_PATH property introduces a custom PID path for the Ranger Admin Process. To configure this property to start and stop of the Ranger Admin service, add the following property to `install.properties`. The default value is `/var/run/ranger`.

```
RANGER_PID_DIR_PATH=/var/run/ranger
```

9. Install the Ranger Admin component. Be sure to set the JAVA_HOME environment variable if it is not already set.

```
cd /usr/hdp/current/ranger-admin/

cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-site.xml ews/webapp/
WEB-INF/classes/conf/

cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-default-site.xml ews/
webapp/WEB-INF/classes/conf/

cp  ews/webapp/WEB-INF/classes/conf.dist/security-applicationContext.xml
 ews/webapp/WEB-INF/classes/conf/

./setup.sh
```

This should successfully install the Ranger Admin component.

10. 

Note

Confirm that `/usr/bin/ranger-admin` points to latest `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh` OR to `/usr/hdp/current/ranger-admin/ews/ranger-admin-services.sh`, where `current` points to the <latest version>.

Start the Ranger Admin component.

```
service ranger-admin start
```

11. Configure and setup the Ranger UserSync component by migrating the properties from the HDP 2.5 `install.properties` file (POLICY_MGR_URL, SYNC_SOURCE and LDAP/AD properties.

Table 1.5. Ranger_Usersync install.properties names and values

Property Name	Property Value
POLICY_MGR_URL	http://<hostname>:6080
SYNC_SOURCE	unix
SYNC_INTERVAL	5

12. Add the following property to the `install.properties` file to set the base directory for the Ranger Usersync process:

```
ranger_base_dir =/etc/ranger
```

13. Add the following properties to the `install.properties` file to set SSL configurations for Ranger Usersync:

```
AUTH_SSL_ENABLED=false
AUTH_SSL_KEYSTORE_FILE=/etc/ranger/usersync/conf/cert/unixauthservice.jks
AUTH_SSL_KEYSTORE_PASSWORD=UnIx529p
AUTH_SSL_TRUSTSTORE_FILE=
AUTH_SSL_TRUSTSTORE_PASSWORD=
```

14. Add the following property to set LDAP delta sync replication:

```
SYNC_LDAP_DELTASYNC = false
```

15. Add the following property to the `install.properties` file to configure the Ranger Usersync PID directory to start and stop the Ranger Usersync service:

```
USERSYNC_PID_DIR_PATH=/var/run/ranger
```

16. Install the Ranger UserSync component. Be sure to set the JAVA_HOME component if it is not already set.

```
cd /usr/hdp/current/ranger-usersync/
cp /usr/hdp/current/ranger-usersync/conf.dist/ranger-ugsync-default.xml /
etc/ranger/usersync/conf
./setup.sh
```

17. Start the Ranger UserSync component.

```
service ranger-usersync start
```

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

Confirm that `/usr/bin/ranger-usersync` points to the latest `/usr/hdp/<version>/ranger-usersync/ews/ranger-usersync-services.sh` OR to `/usr/hdp/current/ranger-usersync/ranger-usersync-services.sh`, where current points to the <latest version>.

1.26.6. Install the Ranger Components

Next, you need to re-install each Ranger component again to ensure you have the latest version. Because you have already upgraded your HDP stack, you only need to follow the instructions in the *Non-Ambari Cluster Installation Guide* to install each Ranger component. You must install the following Ranger components:

- Ranger Policy Admin
- Ranger UserSync
- Ranger Plugins:
  - HDFS
  - HBase
  - Hive
  - Knox
  - Storm
  - Solr
  - Kafka
  - YARN

Note

When installing each Ranger component, you also need to make sure you upgrade each individual component to version 2.6.0 before restarting each service.

1.26.7. Restart the Ranger Services

Once you have re-installed each Ranger component, you then need to restart these components to ensure the new configurations are loaded in your cluster. This includes restarting the Policy Admin and UserSync components, NameNode, and each Ranger plugin.

Note

Before restarting the NameNode, make sure to remove the `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable the NameNode after finishing the upgrade.

The *Non-Ambari Cluster Installation Guide* describes how you can start the following Ranger services:

- Ranger Policy Admin service

```
service ranger-admin start
```

- Ranger UserSync service

```
service ranger-usersync start
```

1.26.8. Enable Ranger Plugins

The final step in the Ranger upgrade process requires you to re-enable the Ranger plugins. Although you are only required to enable HDFS in your cluster, you should re-enable all of the Ranger plugins because class names have changed for the 2.6.0 release, and to ensure smooth operation of Ranger services in your cluster.

Note

When you enable each Ranger plugin, be sure to remove all 2.5 class name values.

Note

Re-enabling a Ranger plugin does not affect policies you have already created. As long as you use the same database as the Policy store, all of your data remains intact.

To re-enable the Ranger plugins, use the links listed below to access instructions in the *Non-Ambari Cluster Installation* guide that describe editing the `install.properties` file and enabling the Ranger plugins:

Important

Before enabling the HDFS plugin, remove `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable this plugin after the upgrade is complete.

- HDFS Plugin
- YARN Plugin
- Kafka Plugin
- HBase Plugin
- Hive Plugin
- Knox Plugin
- Storm Plugin

1.26.9. Enable KMS Configuration

Configure and setup the Ranger KMS configuration by editing the KMS `install.properties` file:

Table 1.6. KMS install.properties names and values

Property Name	Property Name
DB_FLAVOR	MYSQL
db_root_user	foot
db_root_password	<db password>
db_host	<db hostname>
db_name	rangerkms (default)
db_user	rangerkms (default)
db_password	<kms db password>
hadoop_conf	/etc/hadoop/conf
POLICY_MGR_URL	http://<hostname>:6080
REPOSITORY_NAME	kmsdev
XAAUDIT.SOLR.ENABLE	false (default), change to true if desired
XAAUDIT.SOLR.URL	http://<solr_host>:6083/solr/ranger_audits

1.26.10. Configure and Start Apache Ranger on a Kerberized Cluster

Beginning with HDP 2.6.0, kerberos authentication is supported for Ranger and its plugins. Use the this section if you have an HDP 2.5 cluster in a kerberized environment with Ranger in simple authentication mode that you want to upgrade.

For additional information regarding Kerberos, refer to Kerberos Overview in the *Hadoop Security Guide*.

1.26.10.1. Create Keytabs and Principals

1. Follow these steps to create keytabs and principals:

   For Ranger Admin:

   a. Create rangeradmin/<FQDN of Ranger Admin>@<REALM>.

   b.
   ```
   > kadmin.local
   > addprinc -randkey rangeradmin/<FQDN of Ranger Admin>
   Eg: addprinc -randkey rangeradmin/ranger-upgrade-0707-2.openstacklocal
   > xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/<FQDN of
    Ranger Admin>@<REALM>
   Eg: xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/ranger-
   upgrade-0707-2.openstacklocal/EXAMPLE.COM
   >exit
   ```

   For Ranger Lookup:

   a. Create rangerlookup/<FQDN of Ranger Admin>@<REALM>.

   b.
   ```
   > kadmin.local
   > addprinc -randkey  rangerlookup/<FQDN of Ranger Admin>
   Eg: addprinc -randkey rangerlookup/ranger-upgrade-0707-2.openstacklocal
   ```

```
> xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/<FQDN of
 Ranger  Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/ranger-
upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger Usersync:

a. Create rangerusersync/<FQDN>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey rangerusersync/<FQDN of Ranger usersync>
Eg: addprinc -randkey rangerusersync/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger Tagsync

a. Create rangertagsync/<FQDN>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey rangertagsync/<FQDN of Ranger tagsync>
Eg: addprinc -randkey rangertagsync/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger KMS:

a. Create rangerkms/<FQDN of Ranger Admin>@<REALM>

b.
```
> kadmin.local
> addprinc -randkey rangerkms/<FQDN of Ranger Admin>
Eg: addprinc -randkey rangerkms/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/<FQDN of Ranger
 Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/ranger-
upgrade-0707-2.openstacklocal/EXAMPLE.COM
> exit
```

2. If the Kerberos server and admin are on different hosts, copy the keytab on the admin host, assign permission to user `ranger`, and change the permissions.

```
scp the <rangeradmin_keytab_file> to <new_path>
chown ranger <rangeradmin_keytab_path>
chmod 400 <rangeradmin_keytab_path>
```

3. Use **kdestroy** to delete the Keberos credentials cache file.

4. Set the following properties and values in the `/etc/hadoop/conf/core-site.xml` file:

Table 1.7. Properties for the /etc/hadoo/conf/core-site.xml file

Property Name	Property Value
fs.defaultFS	hdfs://ranger-upgrade-0707-2.openstacklocal:8020
hadoop.security.authentication	kerberos
hadoop.security.authorization	true
hadoop.security.auth_to_local	RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/ RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*// RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](rangeradmin@EXAMPLE.COM)s/.*/ranger/ RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/ DEFAULT

See Creating Mappings Between Principals and UNIX Usernames in the *Hadoop Security Guide*.

The following is an example of a `core-site.xml` file with the properties set for Kerberos:

```
<configuration>

    <property>
      <name>fs.defaultFS</name>
      <value>hdfs://ranger-upgrade-0707-2.openstacklocal:8020</value>
      <final>true</final>
    </property>

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

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

<property>
      <name>hadoop.security.auth_to_local</name>
      <value>RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/
RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*//
RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](rangeradmin@EXAMPLE.COM)s/.*/ranger/
RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/
DEFAULT</value>
    </property>

</configuration>
```

1.26.10.2. Run Setup Again for Ranger Admin

1. Stop the ranger-admin service.
2. Add the following properties and values to the ranger-admin `install.properties` file:

Table 1.8. Ranger-admin install.properties

Property Name	Property Value
spnego_principal	HTTP/<FQDN_OF_Ranger_Admin_Cluster>@<REALM>
spnego_keytab	<HTTP keytab path>
token_valid	30
cookie_domain	<FQDN_OF_Ranger_Admin_Cluster>
cookie_path	/
admin_principal	rangeradmin/ <FQDN_OF_Ranger_Admin_Cluster>@<REALM>
admin_keytab	<rangeradmin keytab path>
lookup_principal	rangerlookup/ <FQDN_OF_Ranger_Admin_Cluster>@<REALM>
lookup_keytab	<rangerlookup keytab path>
hadoop_conf	/etc/hadoop/conf

3. Execute the `setup.sh` script.
4. Start the ranger-admin service.
5. Stop the ranger-usersync service.
6. Add the following properties and values to the ranger-sync `install.properties` file:

Table 1.9.

Property Name	Property Value
usersync_principal	rangerusersync/<FQDN>@<REALM>
usersyn_keytab	<rangerusersync keytab path>
hadoop_conf	/etc/hadoop/conf

7. Execute the `setup.sh` script.
8. Start the ranger-sync service.
9. Stop the ranger-tagsync service.
10. Add the following properties and values to the ranger-tagsync `install.properties` file:

Table 1.10. Ranger-tagsync install.properties and values

Property Name	Property Value
tagsync_principal	rangertagsync/<FQDN>@<REALM>
tagsync_keytab	<rangertagsync keytab path>
hadoop_conf	/etc/hadoop/conf

11. Execute the `setup.sh` script.
12. Start the ranger-tagsync service.
13. Stop the ranger-kms service.
14. Add the following properties and values to the ranger-kms `install.properties` file:

Table 1.11. Ranger-kms install.properties and values

Property Name	Property Value
kms_principal	rangerkms/<FQDN>@<REALM>
kms_keytab	<rangerkms keytab path>
hadoop_conf	/etc/hadoop/conf

1.26.10.3. Install and Enable the Ranger HDFS Plugin

This section documents how to install and enable the Ranger HDFS plugin. You might want to consider making similar changes for the other Ranger plugins that you are using.

The Ranger HDFS plugin is located at `/usr/hdp/<version>/ranger-hdfs-plugin`.

Follow these steps to install and enable the Ranger HDFS Plugin:

1. Edit the relevant lines in the `install.properties` file:

```
POLICY_MGR_URL=http://<FQDN of ranger admin host>:6080
REPOSITORY_NAME=hadoopdev
Audit info (Solr/HDFS options available)
```

2. Enter the following commands to enable the HDFS plugin:

```
export JAVA_HOME=<JAVA Path>
./enable-hdfs-plugin.sh
```

3. Start and stop the namenode:

```
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"
```

4. In the custom repo configuration file, add the component user, for example, `hdfs`, as a value for each of the following properties:
   - policy.download.auth.users or policy.grantrevoke.auth.users
   - tag.download.auth.users
5. Verify that the plugin communicates with Ranger admin using the **Audit # plugins** tab.
6. Set the following properties in the `hdfs-site.xml` file:

Table 1.12. hdfs-site.xml Property Names and Values

Property Name	Property Value
dfs.permissions.enabled	true

Property Name	Property Value
dfs.permissions.supergroup	hdfs
dfs.block.access.token.enable	true
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.secondary.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos. keytab	/etc/security/keytabs/spnego.service.keytab
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab
dfs.https.port	50470
dfs.namenode.https-address	Example:ip-10-111-59-170.ec2.internal:50470
dfs.datanode.data.dir.perm	750
dfs.cluster.administrators	hdfs
dfs.namenode.kerberos.internal. spnego.principal	${dfs.web.authentication.kerberos.principal}
dfs.secondary.namenode.kerberos. internal.spnego.principal	${dfs.web.authentication.kerberos.principal}

The following is an example of a `hdfs-site.xml` file with the properties set for Kerberos:

```
<property>
     <name>dfs.permissions</name>
     <value>true</value>
     <description> If "true", enable permission checking in
     HDFS. If "false", permission checking is turned
     off, but all other behavior is
     unchanged. Switching from one parameter value to the other does
     not change the mode, owner or group of files or
     directories. </description>
</property>

<property>
     <name>dfs.permissions.supergroup</name>
     <value>hdfs</value>
     <description>The name of the group of
     super-users.</description>
</property>

<property>
     <name>dfs.namenode.handler.count</name>
     <value>100</value>
     <description>Added to grow Queue size so that more
     client connections are allowed</description>
</property>

<property>
     <name>ipc.server.max.response.size</name>
     <value>5242880</value>
</property>

<property>
     <name>dfs.block.access.token.enable</name>
```

```
     <value>true</value>
     <description> If "true", access tokens are used as capabilities
     for accessing datanodes. If "false", no access tokens are checked on
     accessing datanodes. </description>
</property>

<property>
     <name>dfs.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description> Kerberos principal name for the
     NameNode </description>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description>Kerberos principal name for the secondary NameNode.
     </description>
</property>

<property>
     <!--cluster variant -->
     <name>dfs.secondary.http.address</name>
     <value>ip-10-72-235-178.ec2.internal:50090</value>
     <description>Address of secondary namenode web server</description>
</property>

<property>
     <name>dfs.secondary.https.port</name>
     <value>50490</value>
     <description>The https port where secondary-namenode
     binds</description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.principal</name>
     <value>HTTP/_HOST@EXAMPLE.COM</value>
     <description> The HTTP Kerberos principal used by Hadoop-Auth in the
 HTTP endpoint.
     The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP
     SPNEGO specification.
     </description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.keytab</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
     <description>The Kerberos keytab file with the credentials for the HTTP
     Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
     </description>
</property>

<property>
     <name>dfs.datanode.kerberos.principal</name>
     <value>dn/_HOST@EXAMPLE.COM</value>
     <description>
     The Kerberos principal that the DataNode runs as. "_HOST" is replaced
 by the real
     host name.
     </description>
```

```
</property>

<property>
     <name>dfs.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.secondary.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.datanode.keytab.file</name>
     <value>/etc/security/keytabs/dn.service.keytab</value>
     <description>
     The filename of the keytab file for the DataNode.
     </description>
</property>

<property>
     <name>dfs.https.port</name>
     <value>50470</value>
     <description>The https port where namenode
     binds</description>
</property>

<property>
     <name>dfs.https.address</name>
     <value>ip-10-111-59-170.ec2.internal:50470</value>
     <description>The https address where namenode binds</description>
</property>

<property>
     <name>dfs.datanode.data.dir.perm</name>
     <value>750</value>
     <description>The permissions that should be there on
     dfs.data.dir directories. The datanode will not come up if the
     permissions are different on existing dfs.data.dir directories. If
     the directories don't exist, they will be created with this
     permission.</description>
</property>

<property>
     <name>dfs.access.time.precision</name>
     <value>0</value>
     <description>The access time for HDFS file is precise upto this
     value.The default value is 1 hour. Setting a value of 0
     disables access times for HDFS.
     </description>
</property>
```

```
<property>
     <name>dfs.cluster.administrators</name>
     <value> hdfs</value>
     <description>ACL for who all can view the default
     servlets in the HDFS</description>
</property>

<property>
     <name>ipc.server.read.threadpool.size</name>
     <value>5</value>
     <description></description>
</property>

<property>
     <name>dfs.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

7. For Download Policy to be successful, use the Ranger UI to update the service configuration with the following custom properties:

```
policy.download.auth.users=<Component service user>
tag.download.auth.users=<Component service user>(if tag download)
```

8. For Grant/Revoke for Hive and Hbase to be successful, use the Ranger UI to update the service configuration with the following custom property:

```
policy.grantrevoke.auth.users = <Component service user>
```

9. For Test Connection and Resource Lookup to be successful, use the Ranger UI to add lookup user in the permission list of the policies.

1.27. Configuring and Upgrading Apache Spark

Before you can upgrade Apache Spark, you must have first upgraded your HDP components to the latest version (in this case, 2.6.0). This section assumes that you have already upgraded your components for HDP 2.6.0. If you have not already completed these steps, return to Getting Ready to Upgrade and Upgrade 2.5 Components for instructions on how to upgrade your HDP components to 2.6.0.

The upgrade process installs Spark version 1.6. If you want to use Spark version 2, install version 2 after finishing the HDP 2.6 upgrade process. For more information, see Installing and Configuring Apache Spark 2 in the *Command Line Installation Guide*.

To upgrade Spark, start the service and update configurations.

1. Stop the Spark `history-server`. If you are using the Spark `thrift-server`, stop the `thrift-server`.

```
su - spark -c "$SPARK_HOME/sbin/stop-history-server.sh"
su - spark -c "$SPARK_HOME/sbin/stop-thriftserver.sh"
```

2. Remove any reference to `hdp.version` from the Spark configuration files.

   Remove `spark.yarn.services` property from `spark-defaults.conf`.

   Make sure that `spark.history.provider`, if present, is set to `org.apache.spark.deploy.history.FsHistoryProvider` (the default).

3. Restart the `history-server`:

```
su - spark -c "$SPARK_HOME/sbin/start-history-server.sh"
```

4. If you are using the Spark thrift-server, restart the `thrift-server`. See (Optional) Starting the Spark Thrift Server.

5. Validate the Spark installation. As user `spark`, run the examples in the Running Spark Applications in the *Spark Guide*.

For additional configuration information, see the Spark Guide.

1.28. Upgrade Apache Slider

To upgrade Apache Slider, simply upgrade the Slider client.

1. Upgrade Slider client:

```
hdp-select set slider-client 2.6.1.0.0-<version>
slider version
```

1.29. Upgrade Apache Kafka

Upgrade each Apache Kafka node, one at a time.

You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

You can stop one Kafka broker at a time, upgrade the Kafka binaries to HDP 2.6, and start the broker, before stopping the next broker.

Note

If you are willing to accept downtime, you can take all of the brokers down, update the code, and restart all of the brokers. By default, the brokers start with the new protocol.

You can bump the protocol version and restart, at any time after the brokers are upgraded.

1. Shut down the current Kafka daemon, switch to the new version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.5.4.0-2633
su - kafka -c "usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

3. If the upgrade process fails, follow the steps in "Downgrading Kafka" to return to your previous version of Kafka.

1.29.1. Downgrading Kafka

Downgrade each Kafka node one at a time. You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

1. Shut down the current Kafka daemon, switch to the previous version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.6.1.0-2041
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

1.30. Configuring and Upgrading Apache Zeppelin

Note

The **su** commands in this section use `zeppelin` to represent the Zeppelin Service user. If you are using another name for your Apache Zeppelin Service user, you need to substitute your Zeppelin Service user name for `zeppelin` in each of the **su** commands

For HDP 2.6.0, use the Zeppelin version shipped with HDP 2.6.0.

If you have a previous version of Zeppelin, use the following steps to upgrade Zeppelin:

1. Stop the Zeppelin server:

```
su - zeppelin -c “/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh
 stop”
```

2. Migrate configurations from the old Zeppelin configuration directory, `/etc/zeppelin/<old version>` to `/etc/zeppelin/conf`. Update any references to the old version of hdp in the configuration files.

3. Migrate existing notebooks to the new version. Copy the notebook directory recursively from the Zeppelin notebook folder for the previous HDP version, to the Zeppelin notebook folder in the new HDP version:

```
cp -R /usr/hdp/2.5.5.0-157/zeppelin/notebook/ /usr/hdp/2.6.0.3-8/zeppelin/
notebook/
```

4. (Optional, for clusters using Active Directory authentication) The Active Directory class name changed in HDP 2.6.

   In `/usr/hdp/current/zeppelin-server/conf/shiro.ini`, change `org.apache.zeppelin.server.ActiveDirectoryGroupRealm` to `org.apache.zeppelin.realm.ActiveDirectoryGroupRealm`.

5. Start the Zeppelin server:

```
su - zeppelin -c “/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh
 start”
```

6. Validate the Zeppelin installation in one of the following ways:

   - Use the following command:

```
su - zeppelin -c “/usr/hdp/current/zeppelin-server/bin/zeppelin-daemon.sh
 status”
```

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

1.31. Finalize the Upgrade

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

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

1. Verify your file system health before finalizing the upgrade. After you finalize an upgrade, your backup is discarded!

2. As the $HDFS_USER, enter:

```
su - hdfs -c "dfsadmin -finalizeUpgrade"
```

1.32. Migrate the Audit Logs from DB to Solr

Beginning with HDP 2.6.0, audit log to DB support has been removed. If your logs were previously stored on DB, you can migrate the logs to Solr. Refer to Migrating Audit Logs from DB to Solr.

1.33. Install New HDP 2.6.0 Services

- Install new HDP 2.6.0 Services (see the Non-Ambari Cluster Installation Guide):

  - SmartSense: a next generation subscription model that features upgrade and configuration recommendations.

2. Upgrade from HDP 2.4 to HDP 2.6.0 Manually

This chapter provides instructions on how to manually upgrade to HDP 2.6.0 from the HDP 2.4 release. It assumes the existing HDP 2.4.x was also installed manually.

Important

If you installed and manage HDP 2.4 with Ambari, **you must use the Ambari Upgrade Guide** to perform the HDP 2.4 to HDP 2.6.0 upgrade.

These instructions cover the upgrade between two minor releases. If you need to upgrade between two maintenance releases, follow the upgrade instructions in the HDP Release Notes.

Rolling Upgrade involves complex orchestration as well as side-by-side installation. It is too complex for a manual procedure, and is therefore supported only as an Ambari feature. If you wish to perform a Rolling Upgrade, refer to the Ambari Install instructions to install Ambari, then follow the Ambari Rolling Upgrade instructions, see Ambari Upgrade Guide.

The HDP packages for a complete installation of HDP 2.6.0 occupies about 6.5 GB of disk space.

Warning

Until the upgrade is finalized, no HDFS data is deleted from the cluster. Be sure to review your capacity and ensure that you have extra space available during the upgrade window.

The following provides an overview of steps for upgrading to the latest release of HDP 2.6.x from HDP 2.4:

1. Get ready to upgrade
2. Upgrade HDP 2.4 Components and stop all services
3. Use hdp-select to symlink the HDP 2.6.0 components into "current," in place of the former HDP 2.4 components
4. Configure and Start Apache ZooKeeper
5. Configure and Start Hadoop
6. Start HDFS
7. Configure and start YARN/MapReduce
8. Configure and Start Apache HBase
9. Configure and Start Apache Phoenix

10. Configure and Start Apache Accumulo
11. Configure and Start Apache Tez
12. Configure and Start Apache Hive and Apache HCatalog
13. Configure and Start Apache Oozie
14. Configure and Start Apache WebHCat (Templeton)
15. Configure and Start Apache Pig
16. Configure and Start Apache Sqoop
17. Configure and Start Apache Flume
18. Configure and Start Apache Mahout
19. Configure and Start Apache Hue
20. Configure and Start Apache Knox
21. Configure and Start Apache Falcon
22. Configure and Start Apache Storm
23. Configure and Start Apache Ranger
24. Configure and Start Apache Spark
25. Upgrade Apache Slider
26. Upgrade Apache Kafka
27. Finalize the Upgrade
28. Install new HDP 2.6 services

2.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.4 to HDP 2.6 versions and adding the new HDP 2.6 services. These instructions change your configurations.

Note

You must use **kinit** before running the commands as any particular user.

Hardware recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. The HDP packages for a complete installation of HDP 2.6.0 consumes about 6.5 GB of disk space.

The first step is to ensure you keep a backup copy of your HDP 2.4 configurations.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Back up the HDP directories for any hadoop components you have installed.

   The following is a list of all HDP directories:

   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hive-hcatalog/conf`
   - `/etc/hive-webhcat/conf`
   - `/etc/accumulo/conf`
   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/hue/conf`
   - `/etc/knox/conf`
   - `/etc/zookeeper/conf`
   - `/etc/tez/conf`
   - `/etc/storm/conf`
   - `/etc/falcon/conf`
   - /etc/slider/conf/
   - `/etc/ranger/admin/conf, /etc/ranger/usersync/conf` (If Ranger is installed, also take a backup of install.properties for all the plugins, ranger admin & ranger usersync.)
   - Optional - Back up your userlogs directories, `${mapred.local.dir}/userlogs`.

2. Oozie runs a periodic purge on the shared library directory. The purge can delete libraries that are needed by jobs that started before the upgrade began and

that finish after the upgrade. 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 prior to performing the upgrade:

```
<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>
```

3. Stop all long-running applications deployed using Slider:

   ```
   su - yarn "usr/hdp/current/slider-client/bin/slider list"
   ```

   For all applications returned in previous command, run `su - yarn "/usr/hdp/current/slider-client/bin/slider stop <app_name>"`

4. Stop all services (including MapReduce) except HDFS, ZooKeeper, and Ranger, and client applications deployed on HDFS.

   See Stopping HDP Services for more information.

Component	Command
Accumulo	`/usr/hdp/current/accumulo-client/bin$ / usr/hdp/current/accumulo-client/bin/stop-all.sh`
Knox	`cd $GATEWAY_HOME su - knox -c "bin/gateway.sh stop"`
Falcon	`su - falcon "/usr/hdp/current/falcon-server/bin/falcon-stop"`
Oozie	`su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-stop.sh`
WebHCat	`su - webhcat -c "/usr/hdp/hive-webhcat/sbin/webhcat_server.sh stop"`

Hive	Run this command on the Hive Metastore and Hive Server2 host machine:
	
	`ps aux | awk '{print $1,$2}' | grep hive | awk '{print $2}' | xargs kill >/dev/null 2>&1`
	
	Or you can use the following:
	
	`Killall -u hive -s 15 java`

Component	Command
HBase RegionServers	`su - hbase -c "/usr/hdp/current/hbase-regionserver/bin/hbase-daemon.sh --config /etc/hbase/conf stop regionserver"`
HBase Master host machine	`su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh --config /etc/hbase/conf stop master"`

YARN & Mapred Histro	Run this command on all NodeManagers:
	`su - yarn -c "export /usr/hdp/current/hadoop-client/libexec && /usr/hdp/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop nodemanager"`
	Run this command on the History Server host machine:
	`su - mapred -c "export /usr/hdp/current/hadoop-client/libexec && && /usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-jobhistory-daemon.sh --config /etc/hadoop/conf stop historyserver"`
	Run this command on the ResourceManager host machine(s):
	`su - yarn -c "export /usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop resourcemanager"`
	Run this command on the ResourceManager host machine:
	`su -yarn -c "export /usr/hdp/current/hadoop-client/libex && /usr/hdp/current/hadoop-yarn-timelineserver/sbin/yar-daemon.sh --config /etc/hadoop/conf stop timelineserver"`
	Run this command on the YARN Timeline Server node:
	`su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop timelineserver"`

Storm	`storm kill topology-name` `sudo service supervisord stop`
Spark (History server)	`su - spark -c "/usr/hdp/current/spark-client/sbin/stop-history-server.sh"`

5. If you have the Hive component installed, back up the Hive Metastore database.

   The following instructions are provided for your convenience. For the latest backup instructions, see your database documentation.

Table 2.1. Hive Metastore Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sqlsbr	mysql $dbname < $inputfilename.sqlsbr

Database Type	Backup	Restore
	For example: mysqldump hive > /tmp/mydir/backup_hive.sql	For example: mysql hive < /tmp/mydir/backup_hive.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql sbr For example: sudo -u postgres pg_dump hive > /tmp/mydir/backup_hive.sql	sudo -u $username psql $databasename < $inputfilename.sqlsbr For example: sudo -u postgres psql hive < /tmp/mydir/backup_hive.sql
Oracle	Export the database: exp username/password@database full=yes file=output_file.dmp	Import the database: imp username/password@database file=input_file.dmp

6. If you have the Oozie component installed, back up the Oozie metastore database.

   These instructions are provided for your convenience. Check your database documentation for the latest backup instructions.

Table 2.2. Oozie Metastore Database Backup and Restore

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

7. **Optional**: Back up the Hue database.

   The following instructions are provided for your convenience. For the latest backup instructions, please see your database documentation. For database types that are not listed below, follow your vendor-specific instructions.

Table 2.3. Hue Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sqlsbr For example:	mysql $dbname < $inputfilename.sqlsbr For example:

Database Type	Backup	Restore
	mysqldump hue > /tmp/mydir/backup_hue.sql	mysql hue < /tmp/mydir/backup_hue.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql sbr For example: sudo -u postgres pg_dump hue > /tmp/mydir/backup_hue.sql	sudo -u $username psql $databasename < $inputfilename.sqlsbr For example: sudo -u postgres psql hue < /tmp/mydir/backup_hue.sql
Oracle	Connect to the Oracle database using sqlplus. Export the database. For example: exp username/password@database full=yes file=output_file.dmp mysql $dbname < $inputfilename.sqlsbr	Import the database: For example: imp username/password@database file=input_file.dmp
SQLite	/etc/init.d/hue stop su $HUE_USER mkdir ~/hue_backup sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak /etc/init.d/hue start	/etc/init.d/hue stop cd /var/lib/hue mv desktop.db desktop.db.old sqlite3 desktop.db < ~/hue_backup/desktop.bak /etc/init.d/hue start

8. Back up the Knox data/security directory.

   ```
   cp -RL /etc/knox/data/security ~/knox_backup
   ```

9. Save the namespace by executing the following commands:

   ```
   su - hdfs

   hdfs dfsadmin -safemode enter

   hdfs dfsadmin -saveNamespace
   ```

   Note

   In secure mode, you must have Kerberos credentials for the hdfs user.

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

    ```
    su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-old-fsck-1.log"
    ```

    Note

    In secure mode, you must have Kerberos credentials for the hdfs user.

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

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

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

Important

Make sure the namenode is started.

```
su - hdfs -c "hdfs dfs -ls -R / > dfs-old-lsr-1.log"
```

Note

In secure mode you must have Kerberos credentials for the hdfs user.

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

```
su - hdfs dfsadmin -c "-report > dfs-old-report-1.log"
```

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

```
su - hdfs -c "hdfs dfsadmin -report > dfs-old-report-l.log"
```

d. **Optional:** You can copy all or unrecoverable only data storelibext-customer directory in HDFS to a local file system or to a backup instance of HDFS.

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

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

```
su - hdfs -c "hdfs dfsadmin -finalizeUpgrade"
```

Note

In secure mode, you must have Kerberos credentials for the hdfs user.

13. Stop remaining services (HDFS, ZooKeeper, and Ranger).

See Stopping HDP Services for more information.

Component	Command
HDFS	On all DataNotes: If you are running secure cluster, run following command as root: `/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode` Else:

Component	Command

	`su - hdfs -c "usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode"`
	
	If you are not running a highly available HDFS cluster, stop the Secondary NameNode by executing this command on the Secondary NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop secondarynamenode"`
	
	On the NameNode host machine(s)
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop namenode"`
	
	If you are running NameNode HA, stop the ZooKeeper Failover Controllers (ZKFC) by executing this command on the NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-clinent/sbin/hadoop-deamon.sh --config /etc/hadoop/conf stop zkfc"`
	
	If you are running NameNode HA, stop the JournalNodes by executing these command on the JournalNode host machines:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh ==config /etc/hadoop/conf stop journalnode"`

ZooKeeper Host machines	`su - zookeeper "/usr/hdp/current/zookeeper-server/bin/zookeeper-server stop"`

Ranger (XA Secure)	`service ranger-admin stop`
	
	`service ranger-usersync stop`

14. Back up your NameNode metadata.

Note

It's recommended to take a backup of the full `/hadoop.hdfs/namenode` path.

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

The NameNode metadata is stored in a directory specified in the hdfs-site.xml configuration file under the configuration value "dfs.namenode.dir".

For example, if the configuration value is:

```
<property>
  <name>dfs.namenode.name.dir</name>
  <value>/hadoop/hdfs/namenode</value>
</property>
```

Then, the NameNode metadata files are all housed inside the directory `/hadooop.hdfs/namenode`.

b. Store the layoutVersion of the namenode.

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

15. Verify that edit logs in `${dfs.namenode.name.dir}/current/edits*` are empty.

a. Run: `hdfs oev -i ${dfs.namenode.name.dir}/current/edits_inprogress_* -o edits.out`

b. Verify the edits.out file. It should only have OP_START_LOG_SEGMENT transaction. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<EDITS>
<EDITS_VERSION>-56</EDITS_VERSION>
<RECORD>
<OPCODE>OP_START_LOG_SEGMENT</OPCODE>
<DATA>
<TXID>5749</TXID>
</DATA>
</RECORD>
```

c. If edits.out has transactions other than OP_START_LOG_SEGMENT, run the following steps and then verify edit logs are empty.

- Start the existing version NameNode.
- Ensure there is a new FS image file.
- Shut the NameNode down:

```
hdfs dfsadmin – saveNamespace
```

16. Rename or delete any paths that are reserved in the new version of HDFS.

When upgrading to a new version of HDFS, it is necessary to rename or delete any paths that are reserved in the new version of HDFS. If the NameNode encounters a reserved path during upgrade, it prints an error such as the following:

```
/.reserved is a reserved path and .snapshot is a reserved path
component in this version of HDFS.

Please rollback and delete or rename this path, or upgrade with the
-renameReserved key-value pairs option to automatically rename these
paths during upgrade.
```

Specifying `-upgrade -renameReserved` optional key-value pairs causes the NameNode to automatically rename any reserved paths found during startup.

For example, to rename all paths named `.snapshot` to `.my-snapshot` and change paths named `.reserved` to `.my-reserved`, specify `-upgrade -renameReserved .snapshot=.my-snapshot,.reserved=.my-reserved`.

If no key-value pairs are specified with `-renameReserved`, the NameNode then suffixes reserved paths with:

`.<LAYOUT-VERSION>.UPGRADE_RENAMED`

For example: `.snapshot.-51.UPGRADE_RENAMED`.

Note

We recommend that you perform a `-saveNamespace` before renaming paths (running `-saveNamespace` appears in a previous step in this procedure). This is because a data inconsistency can result if an edit log operation refers to the destination of an automatically renamed file.

Also note that running `-renameReserved` renames all applicable existing files in the cluster. This may impact cluster applications.

17. If you are on JDK 1.6, upgrade the JDK on all nodes to JDK 1.7 or JDK 1.8 before upgrading HDP.

2.2. Upgrade HDP 2.4 Components

Important

See the Release Notes for the HDP 2.6.1.0 repo information.

The upgrade process to HDP 2.6.0 involves the following steps.

Select your OS:

RHEL/CentOS/Oracle 6

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.4 components. This command uninstalls the HDP 2.4 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons "bigtop""
   ```

3. Validate that all HDP 2.4 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.4
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/yum.repos.d/hdp.repo
   ```

5. Install the HDP 2.6.0 repo:

   - Download the hdp.repo file:

     ```
     ?????wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/
     updates/2.6.1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

- Confirm the HDP repository is configured.

  ```
  yum repolist
  ```

  You should see something like this. Verify that you have the HDP-2.6.0 directory:

  ```
  Loaded plugins: fastestmirror, security
  Loading mirror speeds from cached hostfile
  * base: mirrors.cat.pdx.edu
  * extras: linux.mirrors.es.net
  * updates: mirrors.usc.edu
  repo id repo namestatus
  HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
  ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

   ```
   yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "hive-webhcat" "hive-hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez"
    "storm" "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark"
    "slider" "hdp_mon_nagios_addons"
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

   ```
   yum list installed | grep HDP-<old.stack.version.number>
   ```

   No component file names should appear in the returned list.

RHEL/CentOS/Oracle 5 (Deprecated)

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.4 components. This command uninstalls the HDP 2.4 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.4 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.4
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/yum.repos.d/hdp.repo
   ```

5. Install the HDP 2.6.0 repo:

- Download the hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.6.
1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
```

- Confirm the HDP repository is configured.

```
yum repolist
```

You should see something like this. Verify that you have the HDP-2.6.0 directory:

```
Loaded plugins: fastestmirror, security
Loading mirror speeds from cached hostfile
* base: mirrors.cat.pdx.edu
* extras: linux.mirrors.es.net
* updates: mirrors.usc.edu
repo id repo namestatus
HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

```
yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie" "collectd"
 "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez" "storm"
 "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
 "hdp_mon_nagios_addons"
```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

```
yum list installed | grep HDP-<old.stack.version.number>
```

No component file names should appear in the returned list.

SLES 11 SP1

1. On all hosts, clean the yum repository.

```
zypper clean -all
```

2. Remove your old HDP 2.4 components. This command uninstalls the HDP 2.4 components. It leaves the user data, and metadata, but removes your configurations:

```
zypper rm "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
 "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
 "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
 "spark*" "slider*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.4 component binaries are uninstalled:

```
yum list installed | grep @HDP2.4
```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/sles11sp1/2.x/updates/2.6.
   1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez"
    "storm" "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark"
    "slider*" "hdp_mon_nagios_addons"
   ```

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

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-client
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

SLES 11 SP3/SP4, SLES 12

1. On all hosts, clean the zypper repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 2.4 components.

   ```
   zypper rm "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*" "gccxml*"
    "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*"
    "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue" "hue-common"
    "hue-shell" "knox*" "ranger*" "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.4 component binaries are uninstalled:

   ```
   zypper search --installed-only --repo HDP-2.4.0.0
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
   updates/2.6.1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "oozie" "collectd" "gccxml"
    "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez" "storm" "falcon"
    "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "spark-python"
    "hdp_mon_nagios_addons "slider*" "hive-webcat" "hive-hcatalog""
   ```

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-clientt
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

Ubuntu 12

1. On all hosts, clean the apt-get repository.

   ```
   apt-get clean -&-all
   ```

2. Remove your old HDP 2.4 components. This command uninstalls the HDP 2.4 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   apt-get remove "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
    "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez.
   *" "storm*" "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue.*"
    "knox*" "spark*" "slider*" "hdp_mon_nagios_addons" --purge
   ```

3. Validate that all HDP 2.4 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.4
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/apt/sources.list.d/hdp.list
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/
   updates/2.6.1.0.0/hdp.list - O /etc/apt/sources.list.d/hdp.list
   ```

6. Run an update:

   ```
   apt-get update
   ```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

   ```
   apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

Debian

1. On all hosts, clean the apt-get repository.

   ```
   apt-get clean -&-all
   ```

2. Remove your old HDP 2.4 components. This command uninstalls the HDP 2.4 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   apt-get remove "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
    "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*"
    "storm*" "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue*" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.4 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.4
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/apt/sources.list.d/hdp.list
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/
   debian<version>/2.x/updates/2.6.1.0.0/hdp.list - O /etc/apt/
   sources.list.d/hdp.list
   ```

6. Run an update:

   ```
   apt-get update
   ```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.4 components:

   ```
   apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

2.3. Symlink Directories with hdp-select

Warning

HDP 2.6.0 installs hdp-select automatically with the installation or upgrade of the first HDP component. If you have not already upgraded ZooKeeper, hdp-select has not been installed.

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.

1. Before you run hdp-select, remove one link:

   ```
   rm /usr/bin/oozie
   ```

2. Run hdp-select set all on your NameNode and all your DataNodes:

   ```
   hdp-select set all 2.6.1.0.0-<$version>
   ```

   For example:

   ```
   /usr/bin/hdp-select set all 2.6.1.0.0-2800
   ```

2.4. Configure and Start Apache ZooKeeper

Tip

If you are running a highly available HDFS cluster, configure and restart Apache ZooKeeper **before** you upgrade HDFS. This best practice lets the upgraded ZKFC work with your primary NameNode and your Standby NameNode.

1. Replace your configuration after upgrading on all the ZooKeeper nodes. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.

2. Start ZooKeeper.

   On all ZooKeeper server host machines, run the following command to start ZooKeeper and the ZKFC:

   ```
   su - zookeeper -c "/usr/hdp/current/zookeeper-server/bin/zookeeper-server
    start"
   ```

2.5. Configure Hadoop

RHEL/CentOS/Oracle Linux

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed in HDP 2.4. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

SLES

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed since HDP 2.4. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

Ubuntu/Debian

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml

2. Paths have changed in HDP 2.6.0. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code does not load, as this is not where lzo is installed.

2.6. Start Hadoop Core

Warning

Before you start HDFS on a highly available HDFS cluster, you must start the ZooKeeper service. If you do not start the ZKFC, there can be failures.

To start HDFS, run commands as the $HDFS_USER.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using

another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Replace your configuration after upgrading on all the HDFS nodes. Replace the HDFS template configuration in `/etc/hdfs/conf`.

2. If you are upgrading from a highly available HDFS cluster configuration, start all JournalNodes. On each JournalNode host, run the following commands:

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

   

   Important

   All JournalNodes must be running when performing the upgrade, rollback, or finalization operations. If any JournalNodes are down when running any such operation, the operation fails.

3. If you are running HDFS on a highly available namenode, you must first start the ZooKeeper service

   

   Note

   Perform this step only if you are on a highly available HDFS cluster.

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

4. Start the NameNode.

   Because the file system version has now changed you must start the NameNode manually.

   On the active NameNode host, run the following commands:

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

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

   

   Note

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

   To check if the Upgrade is in progress, check that the "\previous" directory has been created in the \NameNode and \JournalNode directories. The "\previous" directory contains a snapshot of the data before upgrade.

   In a highly available HDFS cluster configuration, this NameNode does not enter the standby state as usual. Rather, this NameNode immediately enters the active state, perform an upgrade of its local storage directories, and also perform an upgrade of the

shared edit log. At this point, the standby NameNode in the HA pair is still down. It is out of sync with the upgraded active NameNode.

To synchronize the active and standby NameNode, re-establishing HA, re-bootstrap the standby NameNode by running the NameNode with the '-bootstrapStandby' flag. Do NOT start this standby NameNode with the '-upgrade' flag.

```
su - hdfs -c "hdfs namenode -bootstrapStandby -force"
```

The bootstrapStandby command downloads the most recent fsimage from the active NameNode into the $dfs.name.dir directory of the standby NameNode. You can enter that directory to make sure the fsimage has been successfully downloaded. After verifying, start the ZKFailoverController, then start the standby NameNode. You can check the status of both NameNodes using the Web UI.

5. Verify that the NameNode is up and running:

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

6. If you do not have a highly available HDFS cluster configuration (non_HA namenode), start the Secondary NameNode.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster configuration.

   On the Secondary NameNode host machine, run the following commands:

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

7. Verify that the Secondary NameNode is up and running.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster environment.

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

8. Start DataNodes.

   On each of the DataNodes, enter the following command. Note: If you are working on a `non-secure` DataNode, use $HDFS_USER. For a secure DataNode, use root.

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

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

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

10. Verify that NameNode can go out of safe mode.

```
>su - hdfs -c "hdfs dfsadmin -safemode wait"
```

You should see the following result: `Safe mode is OFF`

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

2.7. Verify HDFS Filesystem Health

Analyze if the filesystem is healthy.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

If you have a secure server, you need Kerberos credentials for hdfs user access.

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

   ```
   su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-new-fsck-1.log"
   ```

   You should see feedback that the filesystem under path / is HEALTHY.

2. Run `hdfs` namespace and report.

   a. List directories.

   ```
   su - hdfs -c "hdfs dfs -ls -R / > dfs-new-lsr-1.log"
   ```

   b. Open the `dfs-new-lsr-l.log` and confirm that you can see the file and directory listing in the namespace.

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

   ```
   su - hdfs -c "hdfs dfsadmin -report > dfs-new-report-1.log"
   ```

   d. Open the `dfs-new-report` file and validate the admin report.

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

   The file names are listed below:

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

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

Note

You must do this comparison manually to catch all errors.

4. From the NameNode WebUI, determine if all DataNodes are up and running.

   ```
   http://<namenode>:<namenodeport
   ```

5. If you are on a highly available HDFS cluster, go to the StandbyNameNode web UI to see if all DataNodes are up and running:

   ```
   http://<standbynamenode>:<namenodeport>
   ```

6. If you are **not** on a highly available HDFS cluster, go to the SecondaryNameNode web UI to see if it the secondary node is up and running:

   ```
   http://<secondarynamenode>:<secondarynamenodeport>
   ```

7. Verify that read and write to hdfs works successfully.

   ```
   hdfs dfs -put [input file] [output file]

   hdfs dfs -cat [output file]
   ```

2.8. Configure YARN and MapReduce

After you upgrade Hadoop, complete the following steps to update your configs.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

In secure mode, you must have Kerberos credentials for the hdfs user.

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

   ```
   su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/
   mapreduce/"

   su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/
   hadoop/mapreduce.tar.gz /hdp/apps/2.6.1.0.0-<$version>/
   mapreduce/"

   su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"

   su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<
   $version>/mapreduce"
   ```

```
su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<
$version>/mapreduce/mapreduce.tar.gz"
```

2. Make sure that the following properties are in `/etc/hadoop/conf/mapred-site.xml`:

   - Make sure mapreduce.application.framework.path exists in mapred-site.xml:

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

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

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

Note

If you are planning to use Spark in yarn-client mode, make Spark work in yarn-client mode 2.6.1.0.0-<$version>.

3. Make sure the following property is in `/etc/hadoop/conf/yarn-site.xml`:

```
<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. On secure clusters only, add the following properties to `/etc/hadoop/conf/yarn-site.xml`:

```
 <property>
  <name>yarn.timeline-service.recovery.enabled:</name>
  <value>TRUE</value>
  </property>

  <property>
  <name>yarn.timeline-service.state-store.class: org.apache.hadoop.yarn.
server.timeline.recovery:</name>
  <value>LeveldbTimelineStateStore</value>
  </property>

  <property>
  <name>yarn.timeline-service.leveldb-state-store.path:</name>
  <value><the same as the default of "yarn.timeline-service-leveldb-
timeline-store.path</value>
  </property>
```

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

   - Create the container-executor.cfg file in /usr/hdp/2.6.1.0.0-<version>/hadoop-yarn/bin/container-executor.

   - Insert the following properties:

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

- yarn.nodemanager.linux-container-executor.group - 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 - Comma-separated list of users who can not run container-executor.
- min.user.id - Minimum value of user id. This prevents system users from running container-executor.
- allowed.system.users - Comma-separated list of allowed system users.

- 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 run it:

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

chmod 6050 /usr/hdp/${hdp.version}/hadoop-yarn-server-nodemanager/bin/
container-executor
```

2.9. Start YARN/MapReduce Services

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

Note

The `su` commands in this section use "yarn" to represent the YARN Service user and mapreduce to represent the MAPREDUCE Service user. If you are using another name for these Service users, you need to substitute your Service user name for "yarn" or "mapreduce" in each of the `su` commands.

1. Manually clear the ResourceManager state store.

```
su - yarn -c "yarn resourcemanager -format-state-store"
```

2. Start the ResourceManager on all your ResourceManager hosts.

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

ps -ef | grep -i resourcemanager
```

3. Start the TimelineServer on your TimelineServer host.

```
su - yarn -c "/usr/hdp/current/hadoop-yarn-tinelineserver/sbin/yarn-daemon.
sh start timelineserver"

ps -ef | grep -i timelineserver
```

4. Start the NodeManager on all your NodeManager hosts.

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

ps -ef | grep -i nodemanager
```

5. To start MapReduce, run the following commands:

```
su - mapreduce -c "/usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-
jobhistory-daemon.sh start historyserver"

ps -ef | grep -i jobhistoryserver
```

2.10. Run Hadoop Smoke Tests

To smoke test your Hadoop upgrade, you can run the following MapReduce job as a regular user.

The job uses MapReduce to write 100MB of data into HDFS with RandomWriter

```
hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.
jar
            randomwriter -Dtest.randomwrite.total_bytes=10000000 test-after-
upgrade.
```

You should see messages similar to:

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

MapReduce upgraded successfully. You can now upgrade your other components.

Basic Troubleshooting

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

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

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

Accessing error messages:

1. Access the ApplicationMaster WebUI to view the container logs.

2. At your console logs for MapReduce job, look for a line with this format:

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

3. Select the logs link under ApplicationMaster table. It redirects you to the container logs. Error messages display here.

2.11. Configure and Start Apache HBase

Note

The `su` commands in this section use "hbase" to represent the HBASE Service user. If you are using another name for your HBASE Service user, you need to substitute your HBASE Service user name for "hbase" in each of the `su` commands.

The hbase.bucketcache.percentage.in.combinedcache is removed in HDP 2.6.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.4 needs to be recalculated to attain identical memory allotments in HDP 2.6.0. The L1 LruBlockCache is whatever hfile.block.cache.size is set to and the L2 BucketCache is whatever hbase.bucketcache.size is set to.

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

2. Start services. From root, assuming that $HBASE_USER=hbase:

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

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

3. Check processes.

   ```
   ps -ef | grep -i hmaster

   ps -ef | grep -i hregion
   ```

2.12. Configure Apache Phoenix

To configure Phoenix, complete the following steps:

1. Add the following property to the `/etc/hbase/hbase-site.xml` file on all HBase nodes, the MasterServer, and all RegionServers to prevent deadlocks from occurring during maintenance on global indexes:

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

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

   ```
   <property>
    <name>phoenix.functions.allowUserDefinedFunctions</name>
   ```

```
 <value>true</value>
 <description>enable UDF functions</description>
</property>
```

3. Ensure the client side hbase-site.xml matches the server side configuration.

4. **(Optional)** To use local indexing, set the following three properties. These properties ensure collocation of data table and local index regions:

Note

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

Add the following two properties, if they do not already exist, to the master side configurations:

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

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

Add the following, if it does not already exist, to the RegionServer side configurations:

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

5. If the folder specified in `hbase.tmp.dir` property on `hbase-site.xml` does not exist, create that directory with adequate permissions.

6. Set the following porperty in the hbase-site.xml file for all RegionServers, but not on the client side:

```
<property>
  <name>hbase.rpc.controllerfactory.class</name>
  <value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</
value>
</property>
```

7. Restart the HBase Master and RegionServers.

Configuring Phoenix to Run in a Secure Cluster

Perform the following additional steps to configure Phoenix to run in a secure Hadoop cluster:

1. To link the HBase configuration file with the Phoenix libraries:

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

2. To link the Hadoop configuration file with the Phoenix libraries:

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

Note

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, ignored java.io.IOException:
No FileSystem for scheme: hdfs
```

2.13. Configure and Start Apache Accumulo

Note

The `su` commands in this section use "accumulo" to represent the Accumulo Service user. If you are using another name for your Apache Accumulo Service user, you need to substitute your Accumulo Service user name for "accumulo" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/accumulo/conf` from the template to the conf directory in Accumulo hosts.

2. Start the services:

```
su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` master"

su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` tserver"

su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` gc"

su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` tracer"

su - accumulo -c "/usr/hdp/current/accumulo-master/bin/start-server.sh
 `hostname` monitor"
```

3. Check that the processes are running

```
ps -ef | grep accumulo
```

or visit http://<hostname>:50095 in your browser

2.14. Configure and Start Apache Tez

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

To upgrade Apache Tez:

1. Copy your previously backed-up copy of tez-site.xml into the /etc/tez/conf directory.

2. Upload the Tez tarball to HDFS.

```
su - hdfs
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:hadoop /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 <hdp_version> is the current HDP version, for example 2.6.1.0.0-2800.

3. Edit the tez.lib.uris property in the tez-site.xml file to point to /hdp/apps/ <hdp_version>/tez/tez.tar.gz

```
...
<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, for example 2.6.1.0.0-2800.

4. **Optional** Earlier releases of Tez did not have access control. In the current version of Tez, the default behavior restricts the ability to view the Tez history to only the owner of the job. To retain unrestricted access for non-secure clusters, set tez.am.view-acls set to "*".

5. Change the value of the tez.tez-ui.history-url.base property to the url for the upgraded Tez View. For information on setting up the Tez view, see Deploying the Tez View in the HDP Ambari Views Guide.

6. **Run Tez Smoke Test**

   To smoke test your Tez upgrade, you can run the following Tez Example job as a regular user.

   MRRSleep Tez Example job sleeps for a defined period of time in mapper and reducer

```
hadoop jar /usr/hdp/current/tez-client/tez-tests-<version>.jar mrrsleep -m 1
 -r 1 -mt 100 -rt 100
```

You should see messages similar to:

```
…DAG: State: SUCCEEDED Progress: 100%
DAG completed. FinalState=SUCCEEDED
```

Tez upgraded successfully. You can now upgrade your other components.

2.15. Configure and Start Apache Hive and Apache HCatalog

Note

The `su` commands in this section use "hive" to represent the Hive Service user. If you are using another name for your Hive Service user, you need to substitute your Hive Service user name for "hive" in each of the `su` commands.

1. Prior to starting the upgrade process, set the following in your hive configuration file:

   ```
   datanucleus.autoCreateSchema=false
   ```

2. Copy the jdbc connector jar from OLD_HIVE_HOME/lib to CURRENT_HIVE_HOME/lib.

3. Upgrade the Apache Hive Metastore database schema. Restart the Hive Metastore database and run:

   ```
   su - hive -c "/usr/hdp/current/hive-metastore/bin/schematool -upgradeSchema
    -dbType" <$databaseType>
   ```

   The value for $databaseType can be derby, mysql, oracle, or postgres.

Note

If you are using Postgres 8 and Postgres 9, you should reset the Hive Metastore database owner to <HIVE_USER>:

```
psql -U <POSTGRES_USER> -c
ALTER DATABASE <HIVE-METASTORE-DB-NAME> OWNER TO <HIVE_USER>
```

Note

If you are using Oracle 11, you may see the following error message:

```
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
optimize.mapjoin.mapreduce does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
heapsize does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
server2.enable.impersonation does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
semantic.analyzer.factory.impl does not exist
14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.auto.
convert.sortmerge.join.noconditionaltask does not exist
Metastore connection URL: jdbc:oracle:thin:@//ip-172-31-42-1.ec2.
internal:1521/XE
```

```
Metastore Connection Driver : oracle.jdbc.driver.OracleDriver
Metastore connection User: hiveuser
Starting upgrade metastore schema from version 0.13.0 to 0.14.0
Upgrade script upgrade-0.13.0-to-0.14.0.oracle.sql
Error: ORA-00955: name is already used by an existing object
 (state=42000,code=955)
Warning in pre-upgrade script pre-0-upgrade-0.13.0-to-0.14.0.
oracle.sql: Schema script failed, errorcode 2
Completed upgrade-0.13.0-to-0.14.0.oracle.sql
schemaTool completed
```

You can safely ignore this message. The error is in the pre-upgrade script and can be ignored; the schematool succeeded.

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

   a. Edit the following properties in the hive-site.xml file:

```
<property>
 <name>fs.file.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.file.impl.disable.cache</
description>
</property>

<property>
 <name>fs.hdfs.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.hdfs.impl.disable.cache
 </description>
<property>
```

   b. **Optional:** To enable the Hive buildin authorization mode, make the following changes. If you want to use the advanced authorization provided by Ranger, refer to the Ranger instructions.

      Set the following Hive authorization parameters in the hive-site.xml file:

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

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

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdConfOnlyAuthorizeFactory</value>
</property>
```

      Also set hive.users.in.admin.role to the list of comma-separated users who need to be added to admin role. A user who belongs to the admin role needs to run the "set

role" command before getting the privileges of the admin role, as this role is not in the current roles by default.

Set the following in the hiveserver2-site.xml file.

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

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

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdHiveAuthorizeFactory/value>
</property>
```

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

```
<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>
```

You can further fine-tune your configuration settings based on node hardware specifications, using the HDP utility script.

5. Start Hive Metastore.

   On the Hive Metastore host machine, run the following command:

   ```
   su - hive -c "nohup /usr/hdp/current/hive-metastore/bin/hive
   --service metastore -hiveconf hive.log.file=hivemetastore.log
   >/var/log/hive/hivemetastore.out 2>/var/log/hive/
   hivemetastoreerr.log &"
   ```

6. Start Hive Server2.

   On the Hive Server2 host machine, run the following command:

   ```
   su - hive

   nohup /usr/hdp/current/hive-server2/bin/hiveserver2 -hiveconf
   hive.metastore.uris=" " -hiveconf hive.log.file=hiveserver2.log
   >/var/log/hive/hiveserver2.out 2> /var/log/hive/
   hiveserver2err.log &
   ```

2.16. Configure and Start Apache Oozie

Note

The duration of the Oozie upgrade is dependent on the amount of job history stored in ooziedb. This history must be backed up and restored during the upgrade process. Best practice when planning for upgrade is to backup ooziedb from your production oozie server and restore it to a test or development oozie server. This can help you estimate the time that is be required to upgrade Oozie during your production upgrade.

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and "oozie" to represent the Oozie Service user. If you are using another name for your HDFS Service user or your Oozie Service user, you need to substitute your Service user names for "hdfs" or "oozie" in each of the `su` commands.

Upgrading Apache Oozie is a complex process. Although the instructions are straightforward, set aside a dedicated block of time to upgrade oozie clients and servers.

Perform the following preparation steps on each oozie server host:

1. You must restore oozie-site.xml from your backup to the conf directory on each oozie server and client.
2. Copy the JDBC jar to libext-customer:
   a. Create the /usr/hdp/current/oozie/libext-customer directory.

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

      mkdir libext-customer
      ```

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

      ```
      chmod -R 777 /usr/hdp/current/oozie-server/libext-customer
      ```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/current/hadoop-client/lib/hadoop*lzo*.jar /usr/hdp/
   current/oozie-server/libext-customer

   cp /usr/share/HDP-oozie/ext.zip /usr/hdp/current/oozie-server/
   libext-customer/
   ```

   Also, copy Oozie db jar in libext-customer.

4. If Falcon was also installed and configured before upgrade in HDP 2.4.x, then after upgrade you might also need to do the following:

   ```
   cp /usr/hdp/current/falcon-server/oozie/ext/falcon-oozie-el-
   extension-"jar /usr/hdp/current/oozie-server/libext-customer
   ```

5. Extract share-lib.

   ```
   /usr/hdp/current/oozie/bin/oozie-setup.sh sharelib create -fs
   hdfs://<namenode>:8020
   ```

   To verify that the sharelibs extracted correctly, run the following command:

   ```
   oozie admin -oozie http://<oozie server host address>:11000/
   oozie -shareliblist
   ```

   There should be:

   - Available ShareLib
   - oozie
   - hive
   - distcp
   - hcatalog
   - sqoop
   - mapreduce-streaming
   - pig

   Change the ownershop and permissions of the oozie directory:

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

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

   Add the two zip files from spark-client phone to spark sharelib:

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

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

6. If a previous version of Oozie was created using auto schema creation, run the following SQL query:

   ```
   insert into oozie_sys (name, data) values ('db.version', '2.6');
   ```

7. As the Oozie user (not root), run the upgrade.

   ```
   su - oozie -c "/usr/hdp/current/oozie-server/bin/ooziedb.sh
   upgrade -run"
   ```

8. As root, prepare the Oozie WAR file.

   ```
   chown oozie:oozie /usr/hdp/current/oozie-server/oozie-server/
   conf/server.xml
   ```

```
su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-setup.sh
prepare-war -d /usr/hdp/current/oozie-server/libext-customer"
```

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

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

9. Make sure that following property is added in oozie-log4j.properties:

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

   where ${oozie.instance.id} is determined by oozie, automatically.

10. If you have custom Oozie actions, you must define them in 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>
    <property>
         <name>oozie.service.SparkConfigurationService.spark.configuration</
    name>
         <value>*=/etc/spark/conf/</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>
    ```

11. Configure HTTPS for the Oozie server.

    a. Create a self signed certificate or get certificate from a trusted CA for the Oozie Server

    b. Import the certificate to the client JDK trust store on all client nodes.

    c. In the Ambari Oozie configuration, set the following environment variables in `oozie-env.sh`, adding them if it does not exist:

       ```
       export OOZIE_HTTPS_PORT=11443
       export OOZIE_HTTPS_KEYSTORE_FILE=/home/oozie/.keystore
       export OOZIE_HTTPS_KEYSTORE_PASS=password
       ```

    d. Change `OOZIE_HTTP_PORT={{oozie_server_port}}` to `OOZIE_HTTP_PORT=11000`.

e. Set the `oozie.base.url` to the HTTPS address.

f. Save the configuration, and restart the Oozie components.

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

13. Check processes.

```
ps -ef | grep -i oozie
```

14. When needed, the `/usr/hdp/current/oozie-server/bin/oozied.sh` script with the *`stop`* parameter stops the Oozie server.

2.17. Configure and Start Apache WebHCat

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and webhcat to represent the WebHCat Service user. If you are using another name for these Service users, you need to substitute your Service user name for "hdfs" or "webhcat" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/webhcat/conf` from the template to the conf directory in webhcat hosts.

2. Modify the Apache WebHCat configuration files.

   a. Upload Pig, Hive and Sqoop tarballs to HDFS as the $HDFS_User (in this example, hdfs):

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/pig/pig.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/hive/hive.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/sqoop/sqoop.
tar.gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/pig"

su - hdfs - "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/pig/
pig.tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/hive"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/hive/
hive.tar.gz"
```

```
su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<$version>/
sqoop/sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"
```

b. Update the following properties in the webhcat-site.xml configuration file, as their values have changed:

```
<property>
 <name>templeton.pig.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/pig/pig.tar.gz</value>
</property>

<property>
 <name>templeton.hive.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/hive/hive.tar.gz</value>
</property>

<property>
 <name>templeton.streaming.jar</name>
 <value>hdfs:///hdp/apps/${hdp.version}/mapreduce/
   hadoop-streaming.jar</value>
 <description>The hdfs path to the Hadoop streaming jar file.</
description>
</property>

<property>
 <name>templeton.sqoop.archive</name>
 <value>hdfs:///hdp/apps/${hdp.version}/sqoop/sqoop.tar.gz</value>
 <description>The path to the Sqoop archive.</description>
</property>

<property>
 <name>templeton.sqoop.path</name>
 <value>sqoop.tar.gz/sqoop/bin/sqoop</value>
 <description>The path to the Sqoop executable.</description>
</property>

<property>
 <name>templeton.sqoop.home</name>
 <value>sqoop.tar.gz/sqoop</value>
 <description>The path to the Sqoop home in the exploded archive.
   </description>
</property>
```

Note

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

c. Add the following property if it is not present in webhcat-sitemxml:

```
<property>
 <name>templeton.libjars</name>
 <value>/usr/hdp/current/zookeeper-client/zookeeper.jar,/usr/hdp/current/
hive-client/lib/hive-common.jar</value>
 <description>Jars to add the classpath.</description>
```

```
</property>
```

d. Remove the following obsolete properties from webhcat-site.xml:

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

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

e. Add new proxy users, if needed. In core-site.xml, make sure the following properties are also set to allow WebHCat to impersonate your additional HDP 2.6.0 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 by 'hcat'.

`hadoop.proxyuser.hcat.hosts`

A comma-separated list of the hosts which are allowed to submit requests by 'hcat'.

3. Start WebHCat:

```
su - webhcat -c "/usr/hdp/current/hive-webhcat/sbin/
webhcat_server.sh start"
```

4. Smoke test WebHCat.

a. If you have a non-secure cluster, on the WebHCat host machine, run the following command to check the status of WebHCat server:

```
curl http://$WEBHCAT_HOST_MACHINE:50111/templeton/v1/status
```

You should see the following return status:

```
"status":"ok","version":"v1"
```

b. If you are using a Kerberos secure cluster, run the following command:

```
curl --negotiate -u: http://$WEBHCAT_HOST_MACHINE:50111/
templeton/v1/status
```

You should see the following return status

```
{"status":"ok","version":"v1"}[machine@acme]$
```

2.18. Configure Apache Pig

Replace your configuration after upgrading. Copy /etc/pig/conf from the template to the conf directory in pig hosts.

2.19. Configure and Start Apache Sqoop

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Replace your configuration after upgrading. Copy /etc/sqoop/conf from the template to the conf directory in sqoop hosts.

2. As the HDFS Service user, upload the Apache Sqoop tarball to HDFS.

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.6.1.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/sqoop/sqoop.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chmod 444 /hdp/apps/2.6.1.0.0-<$version>/sqoop/
sqoop.tar.gz"
```

3. If you are using the MySQL database as a source or target, then the MySQL connector jar must be updated to 5.1.29 or later.

   Refer to the MySQL web site for information on updating the MySQL connector jar.

2.20. Configure, Start, and Validate Apache Flume

1. Replace your configuration after upgrading. Copy /etc/flume/conf from the template to the conf directory in Flume hosts.

2. By default, Apache Flume does not start running immediately upon installation.
   To validate your Flume upgrade, replace your default conf/flume.conf with the

provided flume.conf file, restart Flume, and see if the data is flowing by examining the destination.

Use this flume.conf file:

```
#1. Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1

#2.Describe/configure the source
a1.sources.r1.type = seq

#3. Describe the sink
a1.sinks.k1.type = file_roll
a1.sinks.k1.channel = c1
a1.sinks.k1.sink.directory = /tmp/flume

#4. Use a channel which buffers events in memory
a1.channels.c1.type = memory

#5. Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

3. After starting Flume, check /tmp/flume to see if there are any files there. The files should contain simple sequential numbers.

4. After validating, stop Flume and revert changes to flume.conf to prevent your disk from filling up.

2.21. Configure, Start, and Validate Apache Mahout

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

Replace your configuration after upgrading. Copy /etc/mahout/conf from the template to the conf directory in mahout hosts.

To validate Apache Mahout:

1. Create a test user:

```
su - hdfs -c "dfs -put /tmp/sample-test.txt /user/testuser"
```

2. Set up mahout to convert the plain text file sample-test.txt into a sequence file that is in the output directory mahouttest.

```
mahout seqdirectory --input /user/testuser/sample-test.txt --output /user/
 testuser/mahouttest --charset utf-8
```

2.22. Configure and Start Hue

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

For HDP 2.6.0, use the Hue version shipped with HDP 2.6.0. If you have a previous version of Hue, use the following steps to upgrade Hue.

1. Migrate hue.ini setting from your old hue.ini configuration file to new hue.ini configuration file.
2. If you are using the embedded SQLite database, remember to restore your database after upgrade.

   To restore the database from a backup, make sure the destination database is empty before copying (if necessary, rename or remove the current destination database), then copy your backup to the destination database. For example:

   ```
   su - hue
   cd /var/lib/hue
   mv desktop.db desktop.db.old
   sqlite3 desktop.db < ~/hue_backup/desktop.bak
   exit
   ```

3. Synchronize Database

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

4. Start Hue. As a root user, run the following command on the Hue Server:

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

2.23. Configure and Start Apache Knox

When working with the Apache Knox Gateway in your Hadoop cluster, it is important you have the latest version of Knox installed so you can take advantage of new features and enhancements, in addition to ensuring your instance of Knox is in sync with other Hadoop components (e.g. Ranger, Spark, Hive, Hue, etc.) for stability and performance. For example, if you need to upgrade your Hadoop cluster from 2.4 to 2.6, you should also make sure that your individual Hadoop components are also upgraded to the latest version.

HDP enables you to perform a rolling upgrade in 2.2.x and onward. A rolling upgrade means that you can upgrade a component, or the entire Hadoop stack, without losing service, and your users can continue to use the cluster and run jobs with no application

or server downtime. The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL.

The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL. Once the upgrade process is completed, you are up and running with the latest version of Knox on each server you have designated as a Knox server.

Note

In this document, whenever you see a {} with a value inside, this denotes a value you must define.

2.23.1. Upgrade the Knox Gateway

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Knox manually to the latest version. Because "rolling upgrades" are now supported in HDP 2.6.0, it is not important which version of Knox you are currently running, only that you have an instance of the Knox Gateway running.

Note

If you have not already installed Knox, refer to the "Install the Knox RPMs on the Knox Server" section of the *Non-Ambari Cluster Installation Guide* for instructions on how to install and configure the Knox Gateway.

Before upgrading the Knox Gateway, there are a several steps you must follow to ensure your configuration files, settings, and topology files can be copied to the new Knox Gateway instance when the upgrade is complete, which are described below.

1. Back up your existing `conf` directory if you have not already done so.
2. Stop each Knox server if you have not already done so.

   ```
   su -l knox /usr/hdp/{the current Knox version}/knox/bin/gateway.sh stop
   ```

3. Select the HDP server version you are upgrading to after you have stopped each Knox server if you have not already done so.

   ```
   hdp-select set knox-server {the HDP server version}
   ```

4. 

   Note

   The `su` commands in this section use "knox" to represent the Knox Service user. If you are using another name for your Knox Service user, you need to substitute your Knox Service user name for "knox" in each of the `su` commands.

   For HDP 2.6.0, the default paths for Knox change. Upgrade Knox in order to update these paths.

a. Restore the backed up security directory. This places the master secret and keystores back in place for the new deployment.

b. Start the Gateway:

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

c. Unzip your previously saved configuration directory (the `conf` directory you backed up in step 1) into the new */var/log/knox/gateway.conf* directory to import these files.

d. Restart the Knox server to complete the upgrade.

```
su -l knox /usr/hdp/{the new HDP server version}/knox/bin/gateway.sh
 start
```

2.23.2. Verify the Knox Upgrade

To verify the upgrade was successful, follow the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful.

2. Verify you have cluster access using the `LISTSTATUS` WebHDFS API call.

```
curl -ivk -u {user}:{password} https://{knox host}:8443 /gateway/webhdfs/v1/
tmp?op=LISTSTATUS
```

3. Verify the Knox version using the Knox Admin service and Version API.

```
curl -ivk -u {adminuser}:{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

Note

The Admin API requires you to be a member of an Admin group, as specified in the `admin.xml` authorization provider.

When you have verified the Knox upgrade was successful, you can begin using Knox. If, however, the upgrade was unsuccessful, you need to downgrade the Knox Gateway to the previous version. The steps to downgrade the Knox Gateway are described in the next section.

2.23.3. Downgrade the Knox Gateway to the Previous Version

If the Knox Gateway upgrade was unsuccessful, you need to downgrade Knox to the previous version to ensure you have a working Knox Gateway for your cluster. To downgrade Knox, follow the steps listed below.

1. For each server running Knox, stop the server.

```
su -l knox /usr/hdp/{current HDP server version}/knox/bin/gateway.sh stop
```

2. Select the HDP server version you want to use to downgrade your Knox Gateway.

```
hdp-select set knox-server {previous HDP server version}
```

3. Restart the server.

```
su -l knox /usr/hdp/{previous HDP server version}/knox/bin/gateway.sh start
```

2.23.4. Verify the Knox Downgrade Was Successful

When the restart is complete, verify you are running an older version of Knox by following the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful

2. Verify you have cluster access using the `LISTSTATUS`WebHDFS API call.

3. Check the Knox version using the Knox Admin service and Version API using the following command:

```
curl -ivk -u {adminuser}"{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

2.24. Configure and Validate Apache Falcon

Note

In HDP 2.6.0, if authorization is enabled (for example, in the properties file with `*.falcon.security.authorization.enabled=true`") then Access Control List (ACL) is mandated for all entities.

Upgrade Apache Falcon after you have upgraded HDFS, Hive, Oozie, and Pig. Stop Oozie jobs while running Falcon.

1. Replace your configuration after upgrading. Copy /etc/falcon/conf from the template to the conf directory in falcon hosts.

2. Check your Falcon entities. There should be no changes, but in some cases you may need to update your entities post-upgrade.

3. In HDP 2.6.0 for Falcon, TLS is enabled by default. When TLS is enabled, Falcon starts on https://<falcon_host>.15443/. You can disable TLS by adding the following line to the startup.properties file.

   `"*.falcon.enableTLS=false`

4. If Transport Layer Security (TLS) is disabled, check the client.properties file to make sure the property "falcon.uri" is set as follows:

   `falcon.uri=http://<falcon_host>:15000/`

Troubleshooting:

When upgrading Falcon in HDP 2.5 or later, you might encounter the following error when starting the ActiveMQ server:

```
ERROR - [main:] ~ Failed to start ActiveMQ JMS Message Broker. Reason:
java.lang.NegativeArraySizeException (BrokerService:528)
```

If you encounter this error, follow these steps to delete the ActiveMQ history and then restart Falcon. If you want to retain the history, be sure to back up the ActiveMQ history prior to deleting it.

```
cd <ACTIVEMQ_DATA_DIR>
rm -rf ./localhost
cd /usr/hdp/current/falcon-server
su -l <FALCON_USER>
./bin/falcon-stop
./bin/falcon-start
```

2.25. Configure and Start Apache Storm

Note

The `su` commands in this section use "zookeeper" to represent the ZooKeeper Service user. If you are using another name for your ZooKeeper Service user, you need to substitute your ZooKeeper Service user name for "zookeeper" in each of the `su` commands.

Apache Storm is fairly independent of changes to the HDP cluster:

1. Deactivate all running topologies.
2. Delete all states under zookeeper:

   ```
   /usr/hdp/current/zookeeper-client/bin/zkCli.sh (optionally in
   secure environment specify -server zk.server:port)
   ```

3. `rmr /storm`
4. Delete all states under the storm-local directory:

   ```
   rm -rf <value of stormlocal.dir>
   ```

5. Stop Storm Services on the storm node.
6. Update the following configs in `storm.yaml`:

   - `storm.thrift.transport=org.apache.storm.security.auth.SimpleTransportPlugin`
   - `storm.messaging.transport=org.apache.storm.messaging.netty.Context`
   - `nimbus.topology=org.apache.storm.nimbus.DefaultTopologyValidator`
   - `topology.spout.wait.strategy=org.apache.storm.spout.SleepSpoutWaitStrategy`
   - `topology.kryo.factory=org.apache.storm.serialization.DefaultKryoFactory`

- `topology.tuple.serializer=org.apache.storm.serialization.types.ListDelegate`
- `nimbus.authorizer=org.apache.storm.security.suth.authorizer.SimpleACLAuthor` (applicable only in a secure cluster)
- `drpc.authorizer=org.apache.storm.security.auth.authorizer.DRPCSimpleACLAuth` (applicable only in a secure cluster)
- `ui.filter=org.apache.storm.secuity.auth.KerberosPrincipalToLocal` (applicable only in a secure cluster)

7. Stop ZooKeeper Services on the storm node.

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

8. Remove Storm and zookeeper from the storm node and install the HDP 2.6.0 version:
   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum erase storm

     yum erase zookeeper

     yum install storm

     yum install zookeeper
     ```

   - For **SLES**:

     ```
     zypper rm storm

     zypper rm zookeeper

     zypper install storm

     zypper install zookeeper
     ```

   - For **Ubuntu/Debian**:

     ```
     apt-get remove storm --purge

     apt-get remove zookeeper --purge

     apt-get install storm

     apt-get install zookeeper
     ```

9. Replace your configuration after upgrading. Copy `/etc/storm/conf` from the template to the conf directory .
10. Replace your ZooKeeper configuration after upgrading. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.

11. Ensure ZooKeeper is running. On the storm node, run the following command:

```
su - zookeeper -c "source /etc/zookeeper/conf/zookeeper-env.sh; export
 ZOOCFGDIR=/etc/zookeeper/conf; /usr/hdp/current/zookeeper-server/bin/
zkServer.sh start >> $ZOO_LOG_DIR/zoo.out\"
```

where

- $ZOO_LOG_DIR is the directory where ZooKeeper server logs are stored. For example, /var/log/zookeeper.

12. Start nimbus, then supervisor/ui/drpc/logviewer:

```
/usr/hdp/current/storm-nimbus/bin/storm nimbus.
```

13. Start Storm, using a process controller, such as supervisor:

```
su - storm /usr/hdp/current/storm-supervisor/bin/storm
supervisor
```

You can use the same command syntax to start Storm using nimbus/ui and logviewer.

```
su - storm /usr/hdp/current/storm-supervisor/bin/storm nimbus

su - storm /usr/hdp/current/storm-supervisor/bin/storm ui

su - storm /usr/hdp/current/storm-supervisor/bin/storm logviewer

su - storm /usr/hdp/current/storm-supervisor/bin/storm drpc
```

2.26. Configure and Start Apache Ranger

This section describes how to upgrade the Ranger service.

2.26.1. Preparing Your Cluster to Upgrade Ranger

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Apache Ranger manually to the latest version. This section describes the steps you need to follow to prepare your cluster for the Ranger upgrade.

1. Back up the following Ranger configuration directories:
   - Ranger Policy Administration Service

     ```
     /etc/ranger/admin/conf
     ```

   - Ranger UserSync

     ```
     /etc/ranger/usersync/conf
     ```

   - Ranger Plugins:
     - Hadoop

       ```
       /etc/hadoop/conf
       ```

- Hive

  ```
  /etc/hive/conf
  ```

- HBase

  ```
  /etc/hbase/conf
  ```

- Knox

  ```
  /etc/knox/conf
  ```

- Storm

  ```
  /etc/storm/conf
  ```

2. Backup the Ranger Policy and Audit databases. Make sure to take note of the following details in the `install.properties` file:

   - db_host
   - db_name
   - db_user
   - db_password
   - policy manager configuration
   - LDAP directory configuration
   - LDAP settings
   - LDAP AD domain
   - LDAP URL

   ```
   mysqldump -u root -p root ranger > dest_dir/filename.sql
   mysqldump -u root -p root ranger_audit > dest_dir/audit_filename.sql
   ```

2.26.2. Stop the Ranger Services

Now that you have prepared your cluster for the Ranger upgrade, you need to stop the Ranger Admin and Ranger UserSync services. To stop the Ranger services, perform the steps described below.

1. Stop the Ranger Policy Admin service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-admin stop
   ```

2. Stop the Ranger UserSync service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-usersync stop
   ```

3. Stop the applicable services using the Ranger plugin (HDFS, HBase, Hive, Knox, Storm).

See Stopping HDP Services for more information.

2.26.3. Preparing the Cluster for Upgrade

Before you begin the upgrade process, you need to perform a series of steps to prepare the cluster for upgrade. These steps are described in the "*Getting Ready To Upgrade*" section of this guide, which you need to follow before continuing to upgrade Ranger. Some of these steps include:

- Backing up HDP directories
- Stopping all long-running applications and services.
- Backing up the Hive and Oozie metastore databases.
- Backing up Hue
- Backing up specific directories and configurations

2.26.4. Registering the HDP 2.6.0 Repo

After you have prepared your cluster for the upgrade, you need to register the HDP 2.6.0 repo. This requires you to perform the following steps:

1. (Optional) The Ranger components should already have installed in the at the beginning of the HDP upgrade process, but you can use the following commands to confirm that the Ranger packages have been installed:

   ```
   hdp-select status ranger-admin
   hdp-select status ranger-usersync
   ```

   If the packages have not been installed, you can use the install commands specific to your OS. For example, for RHEL/CentOS you would use the following commands to install the packages.

   ```
   yum install ranger_2_5_*-admin
   yum install ranger_2_5_*-usersync
   ```

2. Select the Ranger Admin and Ranger UserSync versions you want to use.

   ```
   hdp-select set ranger-admin <HDP_server_version>
   hdp-select set ranger-usersync <HDP_server_version>
   ```

3. Change ownership of `/etc/ranger/admin/conf/` and `/etc/ranger/usersync/conf/` to `ranger:ranger`:

   ```
   chown -R ranger:ranger /etc/ranger/admin/conf/
   ```

4. Solr must be installed and configured before installing RangerAdmin or any of the Ranger component plugins.

   For information regarding installation and configuration of Solr, see Using Apache Solr for Ranger Audits.

5. Update the `install.properties` file to migrate the database credentials properties and POLICYMGR_EXTERNAL-URL property from HDP 2.4. to HDP 2.6.0.

Table 2.4. Ranger_Admin install.properties names and values

Property Name	Property Value
DB_FLAVOR	MySql(|ORACLE|POSTGRES|MSSQL|SQLA)
db_root_user	root
db_root_password	Password of db (eg: vagrant)
db_host	Hostname : where your db does exist
policymgr_external_url	http://<hostname>:6080
policymgr_http_enabled	true
authentication_method	UNIX(LDAP|ACTIVE_DIRECTORY|UNIX|NONE)
audit_store	solr
audit_solr_urls	http://<solr_host>:6083/solr/ranger_audits

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

6. Install the Ranger Admin component. Be sure to set the JAVA_HOME environment variable if it is not already set.

```
cd /usr/hdp/current/ranger-admin/

cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-site.xml ews/webapp/
WEB-INF/classes/conf/

cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-default-site.xml ews/
webapp/WEB-INF/classes/conf/

cp  ews/webapp/WEB-INF/classes/conf.dist/security-applicationContext.xml
 ews/webapp/WEB-INF/classes/conf/

./setup.sh
```

This should successfully install the Ranger Admin component.

7. Start the Ranger Admin component.

```
service ranger-admin start
```

8. Configure and setup the Ranger UserSync component by migrating the properties from the HDP 2.4 `install.properties` file (POLICY_MGR_URL, SYNC_SOURCE and LDAP/AD properties.

Table 2.5. Ranger_Usersync install.properties names and values

Property Name	Property Value
POLICY_MGR_URL	http://<hostname>:6080
SYNC_SOURCE	unix

Property Name	Property Value
SYNC_INTERVAL	5

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

9. Install the Ranger UserSync component. Be sure to set the JAVA_HOME component if it is not already set.

```
cd /usr/hdp/current/ranger-usersync/

./setup.sh
```

10. Start the Ranger UserSync component.

```
service ranger-usersync start
```

2.26.5. Install the Ranger Components

Next, you need to re-install each Ranger component again to ensure you have the latest version. Because you have already upgraded your HDP stack, you only need to follow the instructions in the *Non-Ambari Cluster Installation Guide* to install each Ranger component. You must install the following Ranger components:

- Ranger Policy Admin
- Ranger UserSync
- Ranger Plugins:
  - HDFS
  - HBase
  - Hive
  - Knox
  - Storm
  - Solr
  - Kafka
  - YARN

Note

When installing each Ranger component, you also need to make sure you upgrade each individual component to version 2.6.0 before restarting each service.

2.26.6. Restart the Ranger Services

Once you have re-installed each Ranger component, you then need to restart these components to ensure the new configurations are loaded in your cluster. This includes restarting the Policy Admin and UserSync components, NameNode, and each Ranger plugin.

Note

Before restarting the NameNode, make sure to remove the `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable the NameNode after finishing the upgrade.

The *Non-Ambari Cluster Installation Guide* describes how you can start the following Ranger services:

- Ranger Policy Admin service

```
service ranger-admin start
```

- Ranger UserSync service

```
service ranger-usersync start
```

2.26.7. Enable Ranger Plugins

The final step in the Ranger upgrade process requires you to re-enable the Ranger plugins. Although you are only required to enable HDFS in your cluster, you should re-enable all of the Ranger plugins because class names have changed for the 2.6.0 release, and to ensure smooth operation of Ranger services in your cluster.

Note

When you enable each Ranger plugin, be sure to remove all 2.4 class name values.

Note

Re-enabling a Ranger plugin does not affect policies you have already created. As long as you use the same database as the Policy store, all of your data remains intact.

To re-enable the Ranger plugins, use the links listed below to access instructions in the *Non-Ambari Cluster Installation* guide that describe editing the `install.properties` file and enabling the Ranger plugins:

Important

Before enabling the HDFS plugin, remove `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable this plugin after the upgrade is complete.

- HDFS Plugin

- YARN Plugin
- Kafka Plugin
- HBase Plugin
- Hive Plugin
- Knox Plugin
- Storm Plugin

2.26.8. Enable KMS Configuration

Configure and setup the Ranger KMS configuration by editing the KMS `install.properties` file:

Table 2.6. KMS install.properties names and values

Property Name	Property Name
DB_FLAVOR	MYSQL
db_root_user	foot
db_root_password	<db password>
db_host	<db hostname>
db_name	rangerkms (default)
db_user	rangerkms (default)
db_password	<kms db password>
hadoop_conf	/etc/hadoop/conf
POLICY_MGR_URL	http://<hostname>:6080
REPOSITORY_NAME	kmsdev
XAAUDIT.SOLR.ENABLE	false (default), change to true if desired
XAAUDIT.SOLR.URL	http://<solr_host>:6083/solr/ranger_audits

2.26.9. Configure and Start Apache Ranger on a Kerberized Cluster

Beginning with HDP 2.6.0, kerberos authentication is supported for Ranger and its plugins. Use the this section if you have an HDP 2.4 cluster in a kerberized environment with Ranger in simple authentication mode that you want to upgrade.

For additional information regarding Kerberos, refer to Kerberos Overview in the *Hadoop Security Guide*.

2.26.9.1. Create Keytabs and Principals

1. Follow these steps to create keytabs and principals:

   For Ranger Admin:

   a. Create rangeradmin/<FQDN of Ranger Admin>@<REALM>.

   b.
   ```
   > kadmin.local
   ```

```
> addprinc -randkey rangeradmin/<FQDN of Ranger Admin>
Eg: addprinc -randkey rangeradmin/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/<FQDN of
 Ranger Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/ranger-
upgrade-0707-2.openstacklocal/EXAMPLE.COM
>exit
```

For Ranger Lookup:

a. Create rangerlookup/<FQDN of Ranger Admin>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey  rangerlookup/<FQDN of Ranger Admin>
Eg: addprinc -randkey rangerlookup/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/<FQDN of
 Ranger  Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/ranger-
upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger Usersync:

a. Create rangerusersync/<FQDN>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey rangerusersync/<FQDN of Ranger usersync>
Eg: addprinc -randkey rangerusersync/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger Tagsync

a. Create rangertagsync/<FQDN>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey rangertagsync/<FQDN of Ranger tagsync>
Eg: addprinc -randkey rangertagsync/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger KMS:

a. Create rangerkms/<FQDN of Ranger Admin>@<REALM>

b.
```
> kadmin.local
> addprinc -randkey rangerkms/<FQDN of Ranger Admin>
Eg: addprinc -randkey rangerkms/ranger-upgrade-0707-2.openstacklocal
```

```
> xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/<FQDN of Ranger
 Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/ranger-
upgrade-0707-2.openstacklocal/EXAMPLE.COM
> exit
```

2. If the Kerberos server and admin are on different hosts, copy the keytab on the admin host, assign permission to user `ranger`, and change the permissions.

```
scp the <rangeradmin_keytab_file> to <new_path>
chown ranger <rangeradmin_keytab_path>
chmod 400 <rangeradmin_keytab_path>
```

3. Use **kdestroy** to delete the Keberos credentials cache file.

4. Set the following properties and values in the `/etc/hadoop/conf/core-site.xml` file:

Table 2.7. Properties for the /etc/hadoo/conf/core-site.xml file

Property Name	Property Value
fs.defaultFS	hdfs://ranger-upgrade-0707-2.openstacklocal:8020
hadoop.security.authentication	kerberos
hadoop.security.authorization	true
hadoop.security.auth_to_local	RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/ RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*// RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](rangeradmin@EXAMPLE.COM)s/.*/ranger/ RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/ DEFAULT

See Creating Mappings Between Principals and UNIX Usernames in the *Hadoop Security Guide*.

The following is an example of a `core-site.xml` file with the properties set for Kerberos:

```
<configuration>

    <property>
      <name>fs.defaultFS</name>
      <value>hdfs://ranger-upgrade-0707-2.openstacklocal:8020</value>
      <final>true</final>
    </property>

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

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

```
<property>
      <name>hadoop.security.auth_to_local</name>
      <value>RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/
RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*//
RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](rangeradmin@EXAMPLE.COM)s/.*/ranger/
RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/
DEFAULT</value>
    </property>

</configuration>
```

2.26.9.2. Run Setup Again for Ranger Admin

1. Stop the ranger-admin service.
2. Add the following properties and values to the ranger-admin `install.properties` file:

Table 2.8. Ranger-admin install.properties

Property Name	Property Value
spnego_principal	HTTP/<FQDN_OF_Ranger_Admin_Cluster>@<REALM>
spnego_keytab	<HTTP keytab path>
token_valid	30
cookie_domain	<FQDN_OF_Ranger_Admin_Cluster>
cookie_path	/
admin_principal	rangeradmin/ <FQDN_OF_Ranger_Admin_Cluster>@<REALM>
admin_keytab	<rangeradmin keytab path>
lookup_principal	rangerlookup/ <FQDN_OF_Ranger_Admin_Cluster>@<REALM>
lookup_keytab	<rangerlookup keytab path>
hadoop_conf	/etc/hadoop/conf

3. Execute the `setup.sh` script.
4. Start the ranger-admin service.
5. Stop the ranger-usersync service.
6. Add the following properties and values to the ranger-sync `install.properties` file:

Table 2.9.

Property Name	Property Value
usersync_principal	rangerusersync/<FQDN>@<REALM>
usersyn_keytab	<rangerusersync keytab path>
hadoop_conf	/etc/hadoop/conf

7. Execute the `setup.sh` script.
8. Start the ranger-sync service.

9. Stop the ranger-tagsync service.
10. Add the following properties and values to the ranger-tagsync `install.properties` file:

Table 2.10. Ranger-tagsync install.properties and values

Property Name	Property Value
tagsync_principal	rangertagsync/<FQDN>@<REALM>
tagsync_keytab	<rangertagsync keytab path>
hadoop_conf	/etc/hadoop/conf

11. Execute the `setup.sh` script.
12. Start the ranger-tagsync service.
13. Stop the ranger-kms service.
14. Add the following properties and values to the ranger-kms `install.properties` file:

Table 2.11. Ranger-kms install.properties and values

Property Name	Property Value
kms_principal	rangerkms/<FQDN>@<REALM>
kms_keytab	<rangerkms keytab path>
hadoop_conf	/etc/hadoop/conf

2.26.9.3. Install and Enable the Ranger HDFS Plugin

This section documents how to install and enable the Ranger HDFS plugin. You might want to consider making similar changes for the other Ranger plugins that you are using.

The Ranger HDFS plugin is located at `/usr/hdp/<version>/ranger-hdfs-plugin`.

Follow these steps to install and enable the Ranger HDFS Plugin:

1. Edit the relevant lines in the `install.properties` file:

```
POLICY_MGR_URL=http://<FQDN of ranger admin host>:6080
REPOSITORY_NAME=hadoopdev
Audit info (Solr/HDFS options available)
```

2. Enter the following commands to enable the HDFS plugin:

```
export JAVA_HOME=<JAVA Path>
./enable-hdfs-plugin.sh
```

3. Start and stop the namenode:

```
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"
```

4. In the custom repo configuration file, add the component user, for example, `hdfs`, as a value for each of the following properties:

- policy.download.auth.users or policy.grantrevoke.auth.users
- tag.download.auth.users

5. Verify that the plugin communicates with Ranger admin using the **Audit # plugins** tab.
6. Set the following properties in the `hdfs-site.xml` file:

Table 2.12. hdfs-site.xml Property Names and Values

Property Name	Property Value
dfs.permissions.enabled	true
dfs.permissions.supergroup	hdfs
dfs.block.access.token.enable	true
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.secondary.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos. keytab	/etc/security/keytabs/spnego.service.keytab
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab
dfs.https.port	50470
dfs.namenode.https-address	Example:ip-10-111-59-170.ec2.internal:50470
dfs.datanode.data.dir.perm	750
dfs.cluster.administrators	hdfs
dfs.namenode.kerberos.internal. spnego.principal	${dfs.web.authentication.kerberos.principal}
dfs.secondary.namenode.kerberos. internal.spnego.principal	${dfs.web.authentication.kerberos.principal}

The following is an example of a `hdfs-site.xml` file with the properties set for Kerberos:

```
<property>
     <name>dfs.permissions</name>
     <value>true</value>
     <description> If "true", enable permission checking in
     HDFS. If "false", permission checking is turned
     off, but all other behavior is
     unchanged. Switching from one parameter value to the other does
     not change the mode, owner or group of files or
     directories. </description>
</property>

<property>
     <name>dfs.permissions.supergroup</name>
     <value>hdfs</value>
     <description>The name of the group of
     super-users.</description>
</property>

<property>
```

```
     <name>dfs.namenode.handler.count</name>
     <value>100</value>
     <description>Added to grow Queue size so that more
     client connections are allowed</description>
</property>

<property>
     <name>ipc.server.max.response.size</name>
     <value>5242880</value>
</property>

<property>
     <name>dfs.block.access.token.enable</name>
     <value>true</value>
     <description> If "true", access tokens are used as capabilities
     for accessing datanodes. If "false", no access tokens are checked on
     accessing datanodes. </description>
</property>

<property>
     <name>dfs.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description> Kerberos principal name for the
     NameNode </description>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description>Kerberos principal name for the secondary NameNode.
     </description>
</property>

<property>
     <!--cluster variant -->
     <name>dfs.secondary.http.address</name>
     <value>ip-10-72-235-178.ec2.internal:50090</value>
     <description>Address of secondary namenode web server</description>
</property>

<property>
     <name>dfs.secondary.https.port</name>
     <value>50490</value>
     <description>The https port where secondary-namenode
     binds</description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.principal</name>
     <value>HTTP/_HOST@EXAMPLE.COM</value>
     <description> The HTTP Kerberos principal used by Hadoop-Auth in the
 HTTP endpoint.
     The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP
     SPNEGO specification.
     </description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.keytab</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
```

```
     <description>The Kerberos keytab file with the credentials for the HTTP
     Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
     </description>
</property>

<property>
     <name>dfs.datanode.kerberos.principal</name>
     <value>dn/_HOST@EXAMPLE.COM</value>
     <description>
     The Kerberos principal that the DataNode runs as. "_HOST" is replaced
 by the real
     host name.
     </description>
</property>

<property>
     <name>dfs.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.secondary.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.datanode.keytab.file</name>
     <value>/etc/security/keytabs/dn.service.keytab</value>
     <description>
     The filename of the keytab file for the DataNode.
     </description>
</property>

<property>
     <name>dfs.https.port</name>
     <value>50470</value>
     <description>The https port where namenode
     binds</description>
</property>

<property>
     <name>dfs.https.address</name>
     <value>ip-10-111-59-170.ec2.internal:50470</value>
     <description>The https address where namenode binds</description>
</property>

<property>
     <name>dfs.datanode.data.dir.perm</name>
     <value>750</value>
     <description>The permissions that should be there on
     dfs.data.dir directories. The datanode will not come up if the
     permissions are different on existing dfs.data.dir directories. If
```

```
     the directories don't exist, they will be created with this
     permission.</description>
</property>

<property>
     <name>dfs.access.time.precision</name>
     <value>0</value>
     <description>The access time for HDFS file is precise upto this
     value.The default value is 1 hour. Setting a value of 0
     disables access times for HDFS.
     </description>
</property>

<property>
     <name>dfs.cluster.administrators</name>
     <value> hdfs</value>
     <description>ACL for who all can view the default
     servlets in the HDFS</description>
</property>

<property>
     <name>ipc.server.read.threadpool.size</name>
     <value>5</value>
     <description></description>
</property>

<property>
     <name>dfs.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

7. For Download Policy to be successful, use the Ranger UI to update the service configuration with the following custom properties:

```
policy.download.auth.users=<Component service user>
tag.download.auth.users=<Component service user>(if tag download)
```

8. For Grant/Revoke for Hive and Hbase to be successful, use the Ranger UI to update the service configuration with the following custom property:

```
policy.grantrevoke.auth.users = <Component service user>
```

9. For Test Connection and Resource Lookup to be successful, use the Ranger UI to add lookup user in the permission list of the policies.

2.27. Configuring and Upgrading Apache Spark

Before you can upgrade Apache Spark, you must have first upgraded your HDP components to the latest version (in this case, 2.6.0). This section assumes that you have already upgraded your components for HDP 2.6.0. If you have not already completed these steps, return to Getting Ready to Upgrade and Upgrade 2.4 Components for instructions on how to upgrade your HDP components to 2.6.0.

The upgrade process installs Spark version 1.6. If you want to use Spark version 2, install version 2 after finishing the HDP 2.6 upgrade process. For more information, see Installing and Configuring Apache Spark 2 in the *Command Line Installation Guide*.

To upgrade Spark, start the service and update configurations.

1. Stop the Spark `history-server`. If you are using the Spark `thrift-server`, stop the `thrift-server`.

```
su - spark -c "$SPARK_HOME/sbin/stop-history-server.sh"
su - spark -c "$SPARK_HOME/sbin/stop-thriftserver.sh"
```

2. Remove any reference to `hdp.version` from the Spark configuration files.

   Remove `spark.yarn.services` property from `spark-defaults.conf`.

   Make sure that `spark.history.provider`, if present, is set to `org.apache.spark.deploy.history.FsHistoryProvider` (the default).

3. Restart the `history-server`:

```
su - spark -c "$SPARK_HOME/sbin/start-history-server.sh"
```

4. If you are using the Spark thrift-server, restart the `thrift-server`. See (Optional) Starting the Spark Thrift Server.

5. Validate the Spark installation. As user `spark`, run the examples in the Running Spark Applications in the *Spark Guide*.

For additional configuration information, see the Spark Guide.

2.28. Upgrade Apache Slider

To upgrade Apache Slider, simply upgrade the Slider client.

1. Upgrade Slider client:

```
hdp-select set slider-client 2.6.1.0.0-<version>
slider version
```

2.29. Upgrade Apache Kafka

Upgrade each Apache Kafka node one at a time.

You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

You can stop one Kafka broker at a time, upgrade the Kafka binaries to HDP 2.6, and start the broker, before stopping the next broker.

Note

If you are willing to accept downtime, you can take all of the brokers down, update the code, and restart all of the brokers. By default, the brokers start with the new protocol.

You can bump the protocol version and restart, at any time after the brokers are upgraded.

The following steps are specific to upgrading to HDP 2.6 from a version lower than HDP 2.5. Follow these steps prior to upgrading Kafka:

1. Add the following properties to the `server.properties` file on all brokers:

```
inter.broker.protocol.version=CURRENT_KAFKA_VERSION
log.message.format.version=CURRENT_KAFKA_VERSION
```

   Examples of CURRENT_KAFKA_VERSION are `0.8.2.0`, `0.9.0.0` or `0.10.0.0`.

2. Upgrade the brokers one at a time: shut down the broker, update the code, and restart it.

3. Once the entire cluster has been upgraded, bump the protocol version by setting inter.broker.protocol.version to `0.10.1.0`.

4. If your previous message format is `0.10.0`, change log.message.format.version to `0.10.1` (this is a no-op as the message format is the same for both 0.10.0 and 0.10.1). If your previous message format version is lower than `0.10.0`, do not change log.message.format.version yet - this parameter should only change once all consumers have been upgraded to `0.10.0.0` or later.

5. Restart the brokers one at a time for the new protocol version to take effect.

6. If log.message.format.version is lower than `0.10.0`, wait until all consumers have been upgraded to `0.10.0` or later. Once all consumers are upgraded to `0.10.0`, change log.message.format.version to `0.10.1` on each broker, and restart each broker one at a time.

Follow these steps to upgrade Kafka:

1. Shut down the current Kafka daemon, switch to the new version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.4.4.0-2633
su - kafka -c "usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

3. If the upgrade process fails, follow the steps in "Downgrading Kafka" to return to your previous version of Kafka.

2.29.1. Downgrading Kafka

Downgrade each Kafka node one at a time. You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

1. Shut down the current Kafka daemon, switch to the previous version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.6.1.0-2041
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

2.30. Finalize the Upgrade

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

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

1. Verify your file system health before finalizing the upgrade. After you finalize an upgrade, your backup is discarded!

2. As the $HDFS_USER, enter:

```
su - hdfs -c "dfsadmin -finalizeUpgrade"
```

2.31. Migrate the Audit Logs from DB to Solr

Beginning with HDP 2.6.0, audit log to DB support has been removed. If your logs were previously stored on DB, you can migrate the logs to Solr. Refer to Migrating Audit Logs from DB to Solr.

2.32. Install New HDP 2.6.0 Services

1. Install new HDP 2.6.0 Services (see the Non-Ambari Cluster Installation Guide):

   - SmartSense: a next generation subscription model that features upgrade and configuration recommendations.

3. Upgrade from HDP 2.3 to HDP 2.6.0 Manually

This chapter provides instructions on how to manually upgrade to HDP 2.6.0 from the HDP 2.3.0 release. It assumes the existing HDP 2.3 was also installed manually.

Important

If you installed and manage HDP 2.3 with Ambari, **you must use the Ambari Upgrade Guide** to perform the the HDP 2.3 to HDP 2.6.0 upgrade.

These instructions cover the upgrade between two minor releases. If you need to upgrade between two maintenance releases, follow the upgrade instructions in the HDP Release Notes.

Rolling Upgrade involves complex orchestration as well as side-by-side installation. It is too complex for a manual procedure, and is therefore supported only as an Ambari feature. If you wish to perform a Rolling Upgrade, refer to the Ambari Install instructions to install Ambari, then follow the Ambari Rolling Upgrade instructions, see Ambari Upgrade Guide.

The HDP packages for a complete installation of HDP 2.6.0 consumes about 6.5 GB of disk space.

Warning

Until the upgrade is finalized, no HDFS data is deleted from the cluster. Be sure to review your capacity and ensure that you have extra space available during the upgrade window.

The following provides an overview of steps for upgrading to the latest release of HDP 2.3.2 from HDP 2.3:

1. Get ready to upgrade
2. Upgrade HDP 2.3 Components and stop all services
3. Use hdp-select to symlink the HDP 2.3.2 components into "current," in place of the former HDP 2.3 components
4. Configure and Start Apache ZooKeeper
5. Configure and Start Hadoop
6. Start HDFS
7. Configure and start YARN/MapReduce
8. Configure and Start Apache HBase
9. Configure and Start Apache Phoenix

10. Configure and Start Apache Accumulo
11. Configure and Start Apache Tez
12. Configure and Start Apache Hive and Apache HCatalog
13. Configure and Start Apache Oozie
14. Configure and Start Apache WebHCat (Templeton)
15. Configure and Start Apache Pig
16. Configure and Start Apache Sqoop
17. Configure and Start Apache Flume
18. Configure and Start Apache Mahout
19. Configure and Start Apache Hue
20. Configure and Start Apache Knox
21. Configure and Start Apache Falcon
22. Configure and Start Apache Storm
23. Configure and Start Apache Ranger
24. Configure and Start Apache Spark
25. Upgrade Apache Slider
26. Upgrade Apache Kafka
27. Finalize the Upgrade
28. Install new HDP 2.6.0 services

3.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.3 to HDP 2.6.0 versions and adding the new HDP 2.6.0 services. These instructions change your configurations.

Note

You must use **kinit** before running the commands as any particular user.

Hardware recommendations

Although there is no single hardware requirement for installing HDP, there are some basic guidelines. The HDP packages for a complete installation of HDP 2.6 consumes about 6.5 GB of disk space.

The first step is to ensure you keep a backup copy of your HDP 2.3 configurations.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Back up the HDP directories for any hadoop components you have installed.

   The following is a list of all HDP directories:

   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hive-hcatalog/conf`
   - `/etc/hive-webhcat/conf`
   - `/etc/accumulo/conf`
   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/hue/conf`
   - `/etc/knox/conf`
   - `/etc/zookeeper/conf`
   - `/etc/tez/conf`
   - `/etc/storm/conf`
   - `/etc/falcon/conf`
   - /etc/slider/conf/
   - `/etc/ranger/admin/conf, /etc/ranger/usersync/conf` (If Ranger is installed, also take a backup of install.properties for all the plugins, ranger admin & ranger usersync.)
   - Optional - Back up your userlogs directories, `${mapred.local.dir}/userlogs`.

2. Oozie runs a periodic purge on the shared library directory. The purge can delete libraries that are needed by jobs that started before the upgrade began and

that finish after the upgrade. 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 prior to performing the upgrade:

```
<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>
```

3. Stop all long-running applications deployed using Slider:

   ```
   su - yarn -c "/usr/hdp/current/slider-client/bin/slider list"
   ```

   For all applications returned in previous command, run `su - yarn "/usr/hdp/current/slider-client/bin/slider stop <app_name>"`

4. Stop all services (including MapReduce) except HDFS, ZooKeeper, and Ranger, and client applications deployed on HDFS.

   See Stopping HDP Services for more information.

Component	Command
Accumulo	`/usr/hdp/current/accumulo-client/bin/stop-all.sh`
Knox	`cd $GATEWAY_HOME su - knox -c "bin/gateway.sh stop"`
Falcon	`su - falcon "/usr/hdp/current/falcon-server/bin/falcon-stop"`
Oozie	`su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-stop.sh`
WebHCat	`su - webhcat -c "/usr/hdp/hive-webhcat/sbin/webhcat_server.sh stop"`

Hive	Run this command on the Hive Metastore and Hive Server2 host machine:
	
	`ps aux | awk '{print $1,$2}' | grep hive | awk '{print $2}' | xargs kill >/dev/null 2>&1`
	
	Or you can use the following:
	
	`Killall -u hive -s 15 java`

Component	Command
HBase RegionServers	`su - hbase -c "/usr/hdp/current/hbase-regionserver/bin/hbase-daemon.sh --config /etc/hbase/conf stop regionserver"`
HBase Master host machine	`su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh --config /etc/hbase/conf stop master"`

YARN & Mapred History	Run this command on all NodeManagers:
	`su - yarn -c "export HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/hadoop-yarn-nodemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop nodemanager"`
	Run this command on the History Server host machine:
	`su - mapred -c "export HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-jobhistory-daemon.sh --config /etc/hadoop/conf stop historyserver"`
	Run this command on the ResourceManager host machine(s):
	`su - yarn -c "export HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-yarn-resourcemanager/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop resourcemanager"`
	Run this command on the ResourceManager host machine:
	`su - yarn -c "export HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/hadoop-yarn-timelineserver/sbin/yar-daemon.sh --config /etc/hadoop/conf stop timelineserver"`
	Run this command on the YARN Timeline Server node:
	`su -l yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop timelineserver"`

Storm	`storm kill topology-name` `sudo service supervisord stop`
Spark (History server)	`su - spark -c "/usr/hdp/current/spark-client/sbin/stop-history-server.sh"`

5. If you have the Hive component installed, back up the Hive Metastore database.

   The following instructions are provided for your convenience. For the latest backup instructions, see your database documentation.

Table 3.1. Hive Metastore Database Backup and Restore

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

6. If you have the Oozie component installed, back up the Oozie metastore database.

   These instructions are provided for your convenience. Check your database documentation for the latest backup instructions.

Table 3.2. Oozie Metastore Database Backup and Restore

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

7. **Optional**: Back up the Hue database.

   The following instructions are provided for your convenience. For the latest backup instructions, please see your database documentation. For database types that are not listed below, follow your vendor-specific instructions.

Table 3.3. Hue Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sqlsbr For example: mysqldump hue > /tmp/mydir/backup_hue.sql	mysql $dbname < $inputfilename.sqlsbr For example: mysql hue < /tmp/mydir/backup_hue.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql sbr For example: sudo -u postgres pg_dump hue > /tmp/mydir/backup_hue.sql	sudo -u $username psql $databasename < $inputfilename.sqlsbr For example: sudo -u postgres psql hue < /tmp/mydir/backup_hue.sql
Oracle	Connect to the Oracle database using sqlplus. Export the database. For example: exp username/password@database full=yes file=output_file.dmp mysql $dbname < $inputfilename.sqlsbr	Import the database: For example: imp username/password@database file=input_file.dmp
SQLite	/etc/init.d/hue stop su $HUE_USER mkdir ~/hue_backup sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak /etc/init.d/hue start	/etc/init.d/hue stop cd /var/lib/hue mv desktop.db desktop.db.old sqlite3 desktop.db < ~/hue_backup/desktop.bak /etc/init.d/hue start

8. Back up the Knox data/security directory.

   ```
   cp -RL /etc/knox/data/security ~/knox_backup
   ```

9. Save the namespace by executing the following commands:

   ```
   su - hdfs

   hdfs dfsadmin -safemode enter

   hdfs dfsadmin -saveNamespace
   ```

   

   Note

   In secure mode, you must have Kerberos credentials for the hdfs user.

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

    ```
    su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-old-fsck-1.log"
    ```

Note

In secure mode, you must have Kerberos credentials for the hdfs user.

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

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

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

    

    Important

    Make sure the namenode is started.

    ```
    su - hdfs -c "hdfs dfs -ls -R / > dfs-old-lsr-1.log"
    ```

    

    Note

    In secure mode you must have Kerberos credentials for the hdfs user.

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

    ```
    su - hdfs dfsadmin -c "-report > dfs-old-report-1.log"
    ```

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

    ```
    su - hdfs -c "hdfs dfsadmin -report > dfs-old-report-l.log"
    ```

    d. **Optional:** You can copy all or unrecoverable only data storelibext-customer directory in HDFS to a local file system or to a backup instance of HDFS.

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

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

    ```
    su - hdfs -c "hdfs dfsadmin -finalizeUpgrade"
    ```

    

    Note

    In secure mode, you must have Kerberos credentials for the hdfs user.

13. Stop remaining services (HDFS, ZooKeeper, and Ranger).

    See Stopping HDP Services for more information.

Component	Command
HDFS	On all DataNotes: If you are running secure cluster, run following command as root:

Component	Command

	`/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode`
	
	Else:
	
	`su - hdfs -c "usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop datanode"`
	
	If you are not running a highly available HDFS cluster, stop the Secondary NameNode by executing this command on the Secondary NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop secondarynamenode"`
	
	On the NameNode host machine(s)
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop namenode"`
	
	If you are running NameNode HA, stop the ZooKeeper Failover Controllers (ZKFC) by executing this command on the NameNode host machine:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-clinent/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop zkfc"`
	
	If you are running NameNode HA, stop the JournalNodes by executing these command on the JournalNode host machines:
	
	`su - hdfs -c "/usr/hdp/current/hadoop-client/sbin/hadoop-daemon.sh ==config /etc/hadoop/conf stop journalnode"`

ZooKeeper Host machines	`su - zookeeper -c "/usr/hdp/current/zookeeper-server/bin/zookeeper-server stop"`

Ranger (XA Secure)	`service ranger-admin stop`
	
	`service ranger-usersync stop`

14. Back up your NameNode metadata.

Note

It's recommended to take a backup of the full `/hadoop.hdfs/namenode` path.

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

The NameNode metadata is stored in a directory specified in the hdfs-site.xml configuration file under the configuration value "dfs.namenode.name.dir".

For example, if the configuration value is:

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

```
  <value>/hadoop/hdfs/namenode</value>
</property>
```

Then, the NameNode metadata files are all housed inside the directory /hadooop.hdfs/namenode.

b. Store the layoutVersion of the namenode.

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

15. Verify that edit logs in ${dfs.namenode.name.dir}/current/edits* are empty.

a. Run: hdfs oev -i ${dfs.namenode.name.dir}/current/edits_inprogress_* -o edits.out

b. Verify the edits.out file. It should only have OP_START_LOG_SEGMENT transaction. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<EDITS>
<EDITS_VERSION>-56</EDITS_VERSION>
<RECORD>
<OPCODE>OP_START_LOG_SEGMENT</OPCODE>
<DATA>
<TXID>5749</TXID>
</DATA>
</RECORD>
```

c. If edits.out has transactions other than OP_START_LOG_SEGMENT, run the following steps and then verify edit logs are empty.

- Start the existing version NameNode.
- Ensure there is a new FS image file.
- Shut the NameNode down:

```
hdfs dfsadmin – saveNamespace
```

16. Rename or delete any paths that are reserved in the new version of HDFS.

When upgrading to a new version of HDFS, it is necessary to rename or delete any paths that are reserved in the new version of HDFS. If the NameNode encounters a reserved path during upgrade, it prints an error such as the following:

```
/.reserved is a reserved path and .snapshot is a reserved path
component in this version of HDFS.

Please rollback and delete or rename this path, or upgrade with the
-renameReserved key-value pairs option to automatically rename these
paths during upgrade.
```

Specifying -upgrade -renameReserved optional key-value pairs causes the NameNode to automatically rename any reserved paths found during startup.

For example, to rename all paths named `.snapshot` to `.my-snapshot` and change paths named `.reserved` to `.my-reserved`, specify `-upgrade -renameReserved .snapshot=.my-snapshot,.reserved=.my-reserved`.

If no key-value pairs are specified with `-renameReserved`, the NameNode then suffixes reserved paths with:

`.<LAYOUT-VERSION>.UPGRADE_RENAMED`

For example: `.snapshot.-51.UPGRADE_RENAMED`.

Note

We recommend that you perform a `-saveNamespace` before renaming paths (running `-saveNamespace` appears in a previous step in this procedure). This is because a data inconsistency can result if an edit log operation refers to the destination of an automatically renamed file.

Also note that running `-renameReserved` renames all applicable existing files in the cluster. This may impact cluster applications.

17. If you are on JDK 1.6, upgrade the JDK on all nodes to JDK 1.7 or JDK 1.8 before upgrading HDP.

3.2. Upgrade HDP 2.3 Components

Important

See the Release Notes for the HDP 2.6.1.0 repo information.

The upgrade process to HDP 2.6 involves the following steps.

Select your OS:

RHEL/CentOS/Oracle 6

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.3 components. This command uninstalls the HDP 2.3 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons "bigtop""
   ```

3. Validate that all HDP 2.3 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.3
   ```

4. Remove your old hdp.repo file:

```
rm /etc/yum.repos.d/hdp.repo
```

5. Install the HDP 2.6 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.6.
     1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - Confirm the HDP repository is configured.

     ```
     yum repolist
     ```

     You should see something like this. Verify that you have the HDP-2.6.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
     ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

   ```
   yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie" "collectd"
    "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider"
    "hdp_mon_nagios_addons"
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

   ```
   yum list installed | grep HDP-<old.stack.version.number>
   ```

   No component file names should appear in the returned list.

RHEL/CentOS/Oracle 5 (Deprecated)

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.3 components. This command uninstalls the HDP 2.3 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
    "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
    "spark*" "slider*" "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.3 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.3
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/yum.repos.d/hdp.repo
   ```

5. Install the HDP 2.6.0 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.6.
     1.0.0/hdp.repo -O /etc/yum.repos.d/hdp.repo
     ```

   - Confirm the HDP repository is configured.

     ```
     yum repolist
     ```

     You should see something like this. Verify that you have the HDP-2.6.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.6.0 Hortonworks Data Platform Version - HDP-2.6.0
     ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

   ```
   yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie" "collectd"
    "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded.

   ```
   yum list installed | grep HDP-<old.stack.version.number>
   ```

   No component file names should appear in the returned list.

SLES 11 SP1

1. On all hosts, clean the yum repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 2.3 components. This command uninstalls the HDP 2.3 components. It leaves the user data, and metadata, but removes your configurations:

```
zypper rm "hadoop*" "webhcat*" "oozie*" "collectd*" "gccxml*" "pig*" "hdfs*"
 "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*" "falcon*" "flume*"
 "phoenix*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell" "knox*"
 "spark*" "slider*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.3 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.3
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/sles11sp1/2.x/updates/2.6.
   1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "hue" "tez"
    "storm" "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark"
    "slider*" "hdp_mon_nagios_addons"
   ```

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

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-client
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

SLES 11 SP3/SP4, and SLES 12 SP 1

1. On all hosts, clean the zypper repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 2.3 components.

   ```
   zypper rm "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*" "gccxml*"
    "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*" "storm*"
   ```

```
 "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue" "hue-common"
 "hue-shell" "knox*" "ranger*" "spark*" "slider*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.3 component binaries are uninstalled:

   ```
   zypper search --installed-only --repo HDP-2.3.6.0
   ```

4. Remove your old hdp.repo file:

   ```
   rm /etc/zypp/repos.d/hdp.repo
   ```

5. Download the HDP 2.6.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
   updates/2.6.1.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

   ```
   zypper install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn"
    "hadoop-mapreduce" "hadoop-client" "openssl" "oozie" "collectd" "gccxml"
    "pig" "sqoop" "zookeeper" "hbase" "hue" "hive" "tez" "storm" "falcon"
    "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "spark-python"
    "hdp_mon_nagios_addons "slider*" "hive-webcat" "hive-hcatalog""
   ```

   ```
   zypper up -r HDP-2.6.0
   ```

   ```
   zypper install oozie-client
   ```

   

   Note

   If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

7. Verify that the components were upgraded. For example, to verify hdfs, hive, and hcatalog:

   ```
   rpm -qa | grep hdfs, && rpm -qa | grep hive && rpm -qa | grep
   hcatalog
   ```

   No component files names should appear in the returned list.

Ubuntu 12

1. On all hosts, clean the apt-get repository.

   ```
   apt-get clean
   ```

2. Remove your old HDP 2.3 components. This command uninstalls the HDP 2.3 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   apt-get remove "hadoop-2-3-*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
    "gccxml*" "pig-2-3-*" "hdfs*" "sqoop-2-3-*" "zookeeper-2-3-*" "hbase-2-3-
   *" "hive-2-3-*" "tez-2-3-*" "storm-2-3-*" "falcon-2-3-*" "flume-2-3-*"
    "phoenix-2-3-*" "accumulo-2-3-*" "mahout" "hue" "knox-2-3-*" "spark*"
    "slider*" "hdp_mon_nagios_addons" --purge
   ```

3. Validate that all HDP 2.3 component binaries are uninstalled:

```
apt --installed list |grep 2.3.
```

4. Remove your old hdp.repo file:

```
rm /etc/apt/sources.list.d/hdp.list
```

5. Download the HDP 2.6.0 hdp.repo file:

```
wget http://public-repo-1.hortonworks.com/HDP/ubuntu12/2.x/
updates/2.6.1.0.0/hdp.list - O /etc/apt/sources.list.d/hdp.list
```

6. Run an update:

```
apt-get update
```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

```
apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
 "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
 "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
 "hdp_mon_nagios_addons"
```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

Debian

1. On all hosts, clean the apt-get repository.

```
apt-get clean -&-all
```

2. Remove your old HDP 2.3 components. This command uninstalls the HDP 2.3 components. It leaves the user data, and metadata, but removes your configurations:

```
apt-get remove "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
 "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*" "tez*"
 "storm*" "falcon*" "flume*" "phoenix*" "accumulo*" "mahout*" "hue*" "knox*"
 "spark*" "slider*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.3 component binaries are uninstalled:

```
yum list installed | grep @HDP2.3
```

4. Remove your old hdp.repo file:

```
rm /etc/apt/sources.list.d/hdp.list
```

5. Download the HDP 2.6.0 hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/
debian<version>/2.x/updates/2.6.1.0.0/hdp.list - O /etc/apt/
sources.list.d/hdp.list
```

6. Run an update:

   ```
   apt-get update
   ```

7. Install the HDP 2.6.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.3 components:

   ```
   apt-get install "hadoop" "hadoop-hdfs" "libhdfs0" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "openssl" "webhcat" "hcatalog" "oozie"
    "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
    "falcon" "flume" "phoenix" "accumulo" "mahout" "knox" "spark" "slider*"
    "hdp_mon_nagios_addons"
   ```

Note

If you installed Apache Ranger, see Upgrade Ranger for more information on the upgrade path.

3.3. Symlink Directories with hdp-select

Warning

HDP 2.6.0 installs hdp-select automatically with the installation or upgrade of the first HDP component. If you have not already upgraded ZooKeeper, hdp-select has not been installed.

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.

1. Before you run hdp-select, remove one link:

   ```
   rm /usr/bin/oozie
   ```

2. Run hdp-select set all on your NameNode and all your DataNodes:

   ```
   hdp-select set all 2.6.1.0.0-<$version>
   ```

   For example:

   ```
   /usr/bin/hdp-select set all 2.6.1.0.0-2800
   ```

3.4. Configure and Start Apache ZooKeeper

Tip

If you are running a highly available HDFS cluster, configure and restart Apache ZooKeeper **before** you upgrade HDFS. This best practice lets the upgraded ZKFC work with your primary NameNode and your Standby NameNode.

1. Replace your configuration after upgrading on all the ZooKeeper nodes. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.

2. Start ZooKeeper.

   On all ZooKeeper server host machines, run the following command to start ZooKeeper and the ZKFC:

   ```
   su - zookeeper -c "/usr/hdp/current/zookeeper-server/bin/zookeeper-server
    start"
   ```

3.5. Configure Hadoop

RHEL/CentOS/Oracle Linux

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed in HDP 2.3. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-
   amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code do not load, as this is not where lzo is installed.

SLES

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml.

2. Paths have changed since HDP 2.3. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-
   amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code do not load, as this is not where lzo is installed.

Ubuntu/Debian

1. Use the HDP Utility script to calculate memory configuration settings. You must update the memory/cpu settings in yarn-site.xml and mapred-site.xml

2. Paths have changed in HDP 2.6.0. Make sure you remove old path specifications from hadoop-env.sh, such as:

   ```
   export JAVA_LIBRARY_PATH=/usr/lib/hadoop/lib/native/Linux-
   amd64-64
   ```

   If you leave these paths in your hadoop-env.sh file, the lzo compression code do not load, as this is not where lzo is installed.

3.6. Start Hadoop Core

Warning

Before you start HDFS on a highly available HDFS cluster, you must start the ZooKeeper service. If you do not start the ZKFC, there can be failures.

To start HDFS, run commands as the $HDFS_USER.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

1. Replace your configuration after upgrading on all the HDFS nodes. Replace the HDFS template configuration in `/etc/hdfs/conf`.

2. If you are upgrading from a highly available HDFS cluster configuration, start all JournalNodes. On each JournalNode host, run the following commands:

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

   

   ### Important

   All JournalNodes must be running when performing the upgrade, rollback, or finalization operations. If any JournalNodes are down when running any such operation, the operation fails.

3. If you are running HDFS on a highly available namenode, you must first start the ZooKeeper service

   

   ### Note

   Perform this step only if you are on a highly available HDFS cluster.

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

4. Start the NameNode.

   Because the file system version has now changed you must start the NameNode manually.

   On the active NameNode host, run the following commands:

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

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

Note

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

To check if the Upgrade is in progress, check that the "\previous" directory has been created in the \NameNode and \JournalNode directories. The "\previous" directory contains a snapshot of the data before upgrade.

In a highly available HDFS cluster configuration, this NameNode does not enter the standby state as usual. Rather, this NameNode immediately enters the active state, perform an upgrade of its local storage directories, and also perform an upgrade of the shared edit log. At this point, the standby NameNode in the HA pair is still down. It is out of sync with the upgraded active NameNode.

To synchronize the active and standby NameNode, re-establishing HA, re-bootstrap the standby NameNode by running the NameNode with the '-bootstrapStandby' flag. Do NOT start this standby NameNode with the '-upgrade' flag.

```
su - hdfs -c "hdfs namenode -bootstrapStandby -force"
```

The bootstrapStandby command downloads the most recent fsimage from the active NameNode into the $dfs.name.dir directory of the standby NameNode. You can enter that directory to make sure the fsimage has been successfully downloaded. After verifying, start the ZKFailoverController, then start the standby NameNode. You can check the status of both NameNodes using the Web UI.

5. Verify that the NameNode is up and running:

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

6. If you do not have a highly available HDFS cluster configuration (non_HA namenode), start the Secondary NameNode.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster configuration.

   On the Secondary NameNode host machine, run the following commands:

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

7. Verify that the Secondary NameNode is up and running.

   

   Note

   Do not perform this step if you have a highly available HDFS cluster environment.

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

8. Start DataNodes.

   On each of the DataNodes, enter the following command. Note: If you are working on a non-secure DataNode, use $HDFS_USER. For a secure DataNode, use root.

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

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

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

10. Verify that NameNode can go out of safe mode.

    ```
    >su - hdfs -c "hdfs dfsadmin -safemode wait"
    ```

    You should see the following result: `Safe mode is OFF`

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

3.7. Verify HDFS Filesystem Health

Analyze if the filesystem is healthy.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

If you have a secure server, you need Kerberos credentials for hdfs user access.

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

   ```
   su - hdfs -c "hdfs fsck / -files -blocks -locations > dfs-new-fsck-1.log"
   ```

   You should see feedback that the filesystem under path / is HEALTHY.

2. Run `hdfs` namespace and report.

   a. List directories.

      ```
      su - hdfs -c "hdfs dfs -ls -R / > dfs-new-lsr-1.log"
      ```

   b. Open the `dfs-new-lsr-l.log` and confirm that you can see the file and directory listing in the namespace.

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

```
su - hdfs -c "hdfs dfsadmin -report > dfs-new-report-1.log"
```

d. Open the `dfs-new-report` file and validate the admin report.

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

   The file names are listed below:

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

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

   

   Note

   You must do this comparison manually to catch all errors.

4. From the NameNode WebUI, determine if all DataNodes are up and running.

   ```
   http://<namenode>:<namenodeport
   ```

5. If you are on a highly available HDFS cluster, go to the StandbyNameNode web UI to see if all DataNodes are up and running:

   ```
   http://<standbynamenode>:<namenodeport>
   ```

6. If you are **not** on a highly available HDFS cluster, go to the SecondaryNameNode web UI to see if it the secondary node is up and running:

   ```
   http://<secondarynamenode>:<secondarynamenodeport>
   ```

7. Verify that read and write to hdfs works successfully.

   ```
   hdfs dfs -put [input file] [output file]
   ```

   ```
   hdfs dfs -cat [output file]
   ```

3.8. Configure YARN and MapReduce

After you upgrade Hadoop, complete the following steps to update your configs.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Important

In secure mode, you must have Kerberos credentials for the hdfs user.

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

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/
mapreduce/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/
hadoop/mapreduce.tar.gz /hdp/apps/2.6.1.0.0-<$version>/
mapreduce/"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<
$version>/mapreduce"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0.0-<
$version>/mapreduce/mapreduce.tar.gz"
```

2. Make sure that the following properties are in `/etc/hadoop/conf/mapred-site.xml`:

   - Make sure mapreduce.application.framework.path exists in mapred-site.xml:

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

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

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

```
 <final>true</final>
</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}.

Note

If you are planning to use Spark in yarn-client mode, make Spark work in yarn-client mode 2.6.1.0.0-<$version>.

3. Make sure the following property is in `/etc/hadoop/conf/yarn-site.xml`:

```
<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. On secure clusters only, add the following properties to `/etc/hadoop/conf/yarn-site.xml`:

```
 <property>
  <name>yarn.timeline-service.recovery.enabled</name>
  <value>TRUE</value>
  </property>

  <property>
  <name>yarn.timeline-service.state-store.class</name>
  <value>org.apache.hadoop.yarn.server.timeline.recovery.
LeveldbTimelineStateStore</value>
  </property>

  <property>
  <name>yarn.timeline-service.leveldb-state-store.path</name>
  <value><the same as the default of "yarn.timeline-service-leveldb-
timeline-store.path</value>
  </property>
```

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

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

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

     - yarn.nodemanager.linux-container-executor.group - 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 - Comma-separated list of users who can not run container-executor.
     - min.user.id - Minimum value of user id. This prevents system users from running container-executor.
     - allowed.system.users - Comma-separated list of allowed system users.
   - 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 run it:

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

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

3.9. Start YARN/MapReduce Services

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

Note

The `su` commands in this section use "yarn" to represent the YARN Service user and mapreduce to represent the MAPREDUCE Service user. If you are using another name for these Service users, you need to substitute your Service user name for "yarn" or "mapreduce" in each of the `su` commands.

1. Manually clear the ResourceManager state store.

   ```
   su - yarn -c "yarn resourcemanager -format-state-store"
   ```

2. Start the ResourceManager on all your ResourceManager hosts.

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

ps -ef | grep -i resourcemanager
```

3. Start the TimelineServer on your TimelineServer host.

```
su - yarn -c "/usr/hdp/current/hadoop-yarn-timelineserver/sbin/yarn-daemon.
sh start timelineserver"

ps -ef | grep -i timelineserver
```

4. Start the NodeManager on all your NodeManager hosts.

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

ps -ef | grep -i nodemanager
```

5. To start MapReduce, run the following commands:

```
su - mapreduce -c "/usr/hdp/current/hadoop-mapreduce-historyserver/sbin/mr-
jobhistory-daemon.sh start historyserver"

ps -ef | grep -i jobhistoryserver
```

3.10. Run Hadoop Smoke Tests

To smoke test your Hadoop upgrade, you can run the following MapReduce job as a regular user.

The job uses MapReduce to write 100MB of data into HDFS with RandomWriter

```
hadoop jar /usr/hdp/current/hadoop-mapreduce-client/hadoop-mapreduce-examples.
jar
            randomwriter -Dtest.randomwrite.total_bytes=10000000 test-after-
upgrade.
```

You should see messages similar to:

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

MapReduce upgraded successfully. You can now upgrade your other components.

Basic Troubleshooting

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

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

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

Accessing error messages:

1. Access the ApplicationMaster WebUI to view the container logs.
2. At your console logs for MapReduce job, look for a line with this format:

   13/10/02 17:57:21 INFO mapreduce.Job: The url to track the job: http://<resource manager host>:8088/proxy/application_1380673658357_0007/
3. Select the logs link under ApplicationMaster table. It redirects you to the container logs. Error messages display here.

3.11. Configure and Start Apache HBase

Note

The `su` commands in this section use "hbase" to represent the HBASE Service user. If you are using another name for your HBASE Service user, you need to substitute your HBASE Service user name for "hbase" in each of the `su` commands.

To enable Kerberos for clusters with dual home network setting, each HBase RegionServer must have its own key. See Installing HDP Manually.

The hbase.bucketcache.percentage.in.combinedcache is removed in HDP 2.6.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.3 needs to be recalculated to attain identical memory allotments in HDP 2.6.0. The L1 LruBlockCache is whatever hfile.block.cache.size is set to and the L2 BucketCache is whatever hbase.bucketcache.size is set to.

1. Replace your configuration after upgrading. Replace the Apache HBase template configuration in `/etc/hbase/conf`.
2. Start services. From root, assuming that $HBASE_USER=hbase:

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

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

   ```
   ps -ef | grep -i hmaster

   ps -ef | grep -i hregion
   ```
4. Run HBase smoke test. Verify that following commands can be executed successfully from HBase shell.

   ```
   create 'testtable', 'cf1'

   put 'testtable', 'r1', 'cf1:c1', 'v1'

   disable 'testtable'
   ```

```
drop 'testtable'
```

3.12. Configure Apache Phoenix

To configure Phoenix, complete the following steps:

1. Add the following property to the `/etc/hbase/hbase-site.xml` file on all HBase nodes, the MasterServer, and all RegionServers to prevent deadlocks from occurring during maintenance on global indexes:

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

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

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

3. Ensure the client side hbase-site.xml matches the server side configuration.

4. **(Optional)** To use local indexing, set the following three properties. These properties ensure collocation of data table and local index regions:

Note

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

Add the following two properties, if they do not already exist, to the master side configurations:

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

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

Add the following, if it does not already exist, to the RegionServer side configurations:

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

```
</property>
```

5. If the folder specified in `hbase.tmp.dir` property on `hbase-site.xml` does not exist, create that directory with adequate permissions.

6. Set the following porperty in the hbase-site.xml file for all RegionServers, but not on the client side:

```
<property>
  <name>hbase.rpc.controllerfactory.class</name>
  <value>org.apache.hadoop.hbase.ipc.controller.ServerRpcControllerFactory</
value>
</property>
```

7. Restart the HBase Master and RegionServers.

Configuring Phoenix to Run in a Secure Cluster

Perform the following additional steps to configure Phoenix to run in a secure Hadoop cluster:

1. To link the HBase configuration file with the Phoenix libraries:

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

2. To link the Hadoop configuration file with the Phoenix libraries:

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

Note

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, ignored java.io.IOException:
No FileSystem for scheme: hdfs
```

3.13. Configure and Start Apache Accumulo

Note

The `su` commands in this section use "accumulo" to represent the Accumulo Service user. If you are using another name for your Apache Accumulo Service user, you need to substitute your Accumulo Service user name for "accumulo" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/accumulo/conf` from the template to the conf directory in Accumulo hosts.

2. Start the services:

```
ssh `hostname` "sudo su - -c \"export ACCUMULO_CONF_DIR=/etc/accumulo/conf/
server;/usr/hdp/current/accumulo-client/bin/start-server.sh `hostname`
 master\" accumulo"

ssh `hostname` "sudo su - -c \"export ACCUMULO_CONF_DIR=/etc/accumulo/conf/
server;/usr/hdp/current/accumulo-client/bin/start-server.sh `hostname`
 tserver\" accumulo"

ssh `hostname` "sudo su - -c \"export ACCUMULO_CONF_DIR=/etc/accumulo/conf/
server;/usr/hdp/current/accumulo-client/bin/start-server.sh `hostname` gc\"
 accumulo"

ssh `hostname` "sudo su - -c \"export ACCUMULO_CONF_DIR=/etc/accumulo/conf/
server;/usr/hdp/current/accumulo-client/bin/start-server.sh `hostname`
 tracer\" accumulo"

ssh `hostname` "sudo su - -c \"export ACCUMULO_CONF_DIR=/etc/accumulo/conf/
server;/usr/hdp/current/accumulo-client/bin/start-server.sh `hostname`
 monitor\" accumulo"
```

3. Check that the processes are running

   ```
   ps -ef | grep accumulo
   ```

   or visit http://<hostname>:50095 in your browser

4. Run Accumulo smoke test. Verify that following commands can be executed successfully from Accumulo shell.

   ```
   createtable testtable

   insert row1 colf colq value1

   scan

   deletetable testtable

   tables
   ```

3.14. Configure and Start Apache Tez

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

To upgrade Apache Tez:

1. Copy your previously backed-up copy of tez-site.xml into the /etc/tez/conf directory.
2. Upload the Tez tarball to HDFS.

```
su - hdfs
```

```
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:hadoop /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 <hdp_version> is the current HDP version, for example 2.6.1.0.0-2800.

3. Edit the tez.lib.uris property in the tez-site.xml file to point to /hdp/apps/ <hdp_version>/tez/tez.tar.gz

```
...
<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, for example 2.6.1.0.0-2800.

4. **Optional** Earlier releases of Tez did not have access control. In the current version of Tez, the default behavior restricts the ability to view the Tez history to only the owner of the job. To retain unrestricted access for non-secure clusters, set tez.am.view-acls set to "*".

5. Change the value of the tez.tez-ui.history-url.base property to the url for the upgraded Tez View. For information on setting up the Tez view, see Deploying the Tez View in the HDP Ambari Views Guide.

6. **Run Tez Smoke Test**

   To smoke test your Tez upgrade, you can run the following Tez Example job as a regular user.

   MRRSleep Tez Example job sleeps for a defined period of time in mapper and reducer

```
hadoop jar /usr/hdp/current/tez-client/tez-tests-<version>.jar mrrsleep -m 1
 -r 1 -mt 100 -rt 100
```

   You should see messages similar to:

```
…DAG: State: SUCCEEDED Progress: 100%
DAG completed. FinalState=SUCCEEDED
```

   Tez upgraded successfully. You can now upgrade your other components.

3.15. Configure and Start Apache Hive and Apache HCatalog

Note

The su commands in this section use "hive" to represent the Hive Service user. If you are using another name for your Hive Service user, you need to substitute your Hive Service user name for "hive" in each of the su commands.

1. Restore backup copy of hive-site.xml to /etc/hive/conf, on all nodes where hivemetastore/hiveserver2 should be running. Prior to starting the upgrade process, set the following in your hive configuration file:

   ```
   datanucleus.autoCreateSchema=false
   ```

2. Copy the jdbc connector jar from OLD_HIVE_HOME/lib to CURRENT_HIVE_HOME/lib.

3. Upgrade the Apache Hive Metastore database schema. Restart the Hive Metastore database and run:

   ```
   su - hive -c "/usr/hdp/current/hive-metastore/bin/schematool -upgradeSchema
    -dbType <$databaseType>"
   ```

   The value for $databaseType can be derby, mysql, oracle, or postgres.

   

   Note

   If you are using Postgres 8 and Postgres 9, you should reset the Hive Metastore database owner to <HIVE_USER>:

   ```
   psql -U <POSTGRES_USER> -c
   ALTER DATABASE <HIVE-METASTORE-DB-NAME> OWNER TO <HIVE_USER>
   ```

   

   Note

   If you are using Oracle 11, you may see the following error message:

   ```
   14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
   optimize.mapjoin.mapreduce does not exist
   14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
   heapsize does not exist
   14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
   server2.enable.impersonation does not exist
   14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.
   semantic.analyzer.factory.impl does not exist
   14/11/17 14:11:38 WARN conf.HiveConf: HiveConf of name hive.auto.
   convert.sortmerge.join.noconditionaltask does not exist
   Metastore connection URL: jdbc:oracle:thin:@//ip-172-31-42-1.ec2.
   internal:1521/XE
   Metastore Connection Driver : oracle.jdbc.driver.OracleDriver
   Metastore connection User: hiveuser
   Starting upgrade metastore schema from version 0.13.0 to 0.14.0
   Upgrade script upgrade-0.13.0-to-0.14.0.oracle.sql
   Error: ORA-00955: name is already used by an existing object
    (state=42000,code=955)
   Warning in pre-upgrade script pre-0-upgrade-0.13.0-to-0.14.0.
   oracle.sql: Schema script failed, errorcode 2
   Completed upgrade-0.13.0-to-0.14.0.oracle.sql
   schemaTool completed
   ```

   You can safely ignore this message. The error is in the pre-upgrade script and can be ignored; the schematool succeeded.

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

   a. Edit the following properties in the hive-site.xml file:

```
<property>
 <name>fs.file.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.file.impl.disable.cache</
description>
</property>

<property>
 <name>fs.hdfs.impl.disable.cache</name>
 <value>false</value>
 <description>Set to false or remove fs.hdfs.impl.disable.cache
 </description>
<property>
```

b. Remove the following property in hive-site.xml, if present:

```
<property>
   <name>hive.exec.post.hooks</name>
   <value>org.apache.hadoop.hive.ql.hooks.ATSHook,org.apache.atlas.hive.
hook.HiveHook</value>
</property>
```

c. **Optional:** To enable the Hive buildin authorization mode, make the following changes. If you want to use the advanced authorization provided by Ranger, refer to the Ranger instructions.

Set the following Hive authorization parameters in the hive-site.xml file:

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

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

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdConfOnlyAuthorizeFactory</value>
</property>
```

Also set hive.users.in.admin.role to the list of comma-separated users who need to be added to admin role. A user who belongs to the admin role needs to run the "set role" command before getting the privileges of the admin role, as this role is not in the current roles by default.

Set the following in the hiveserver2-site.xml file.

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

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

<property>
 <name>hive.security.authorization.manager</name>
 <value>org.apache.hadoop.hive.ql.security.authorization.plugin.sqlstd.
SQLStdHiveAuthorizeFactory/value>
</property>
```

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

```
<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>
```

You can further fine-tune your configuration settings based on node hardware specifications, using the HDP utility script.

5. Start Hive Metastore.

   On the Hive Metastore host machine, run the following command:

   ```
   su - hive -c "nohup /usr/hdp/current/hive-metastore/bin/hive
   --service metastore -hiveconf hive.log.file=hivemetastore.log
   >/var/log/hive/hivemetastore.out 2>/var/log/hive/
   hivemetastoreerr.log &"
   ```

6. Start Hive Server2.

   On the Hive Server2 host machine, run the following command:

   ```
   su - hive

   nohup /usr/hdp/current/hive-server2/bin/hiveserver2 -hiveconf
   hive.metastore.uris=" " -hiveconf hive.log.file=hiveserver2.log
   >/var/log/hive/hiveserver2.out 2> /var/log/hive/
   hiveserver2err.log &
   ```

3.16. Configure and Start Apache Oozie

Note

The duration of the Oozie upgrade is dependent on the amount of job history stored in ooziedb. This history must be backed up and restored during the upgrade process. Best practice when planning for upgrade is to backup ooziedb from your production oozie server and restore it to a test or development oozie

server. This can help you estimate the time that is required to upgrade Oozie during your production upgrade.

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and "oozie" to represent the Oozie Service user. If you are using another name for your HDFS Service user or your Oozie Service user, you need to substitute your Service user names for "hdfs" or "oozie" in each of the `su` commands.

Upgrading Apache Oozie is a complex process. Although the instructions are straightforward, set aside a dedicated block of time to upgrade oozie clients and servers.

Perform the following preparation steps on each oozie server host:

1. You must restore oozie-site.xml from your backup to the conf directory on each oozie server and client.
2. Copy the JDBC jar to libext-customer:
   a. Create the /usr/hdp/current/oozie/libext-customer directory.

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

      mkdir libext-customer
      ```

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

      ```
      chmod -R 777 /usr/hdp/current/oozie-server/libext-customer
      ```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/current/hadoop-client/lib/hadoop*lzo*.jar /usr/hdp/
   current/oozie-server/libext-customer

   cp /usr/share/HDP-oozie/ext.zip /usr/hdp/current/oozie-server/
   libext-customer/
   ```

   Also, copy Oozie db jar in libext-customer.
4. If Falcon was also installed and configured before upgrade in HDP 2.3.x, then after upgrade you might also need to do the following:

   ```
   cp /usr/hdp/current/falcon-server/oozie/ext/falcon-oozie-el-
   extension-*jar /usr/hdp/current/oozie-server/libext-customer
   ```

5. Extract share-lib.

   ```
   /usr/hdp/current/oozie/bin/oozie-setup.sh sharelib create -fs
   hdfs://<namenode>:8020
   ```

   To verify that the sharelibs extracted correctly, run the following command:

   ```
   oozie admin -oozie http://<oozie server host address>:11000/
   oozie -shareliblist
   ```

There should be:

- Available ShareLib
- oozie
- hive
- distcp
- hcatalog
- sqoop
- mapreduce-streaming
- pig

Change the ownershop and permissions of the oozie directory:

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

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

Add the two zip files from spark-client phone to spark sharelib:

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

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

6. If a previous version of Oozie was created using auto schema creation, run the following SQL query:

```
insert into oozie_sys (name, data) values ('db.version', '2.6');
```

7. As the Oozie user (not root), run the upgrade.

```
su - oozie -c "/usr/hdp/current/oozie-server/bin/ooziedb.sh
upgrade -run"
```

8. As root, prepare the Oozie WAR file.

```
chown oozie:oozie /usr/hdp/current/oozie-server/oozie-server/
conf/server.xml

su - oozie -c "/usr/hdp/current/oozie-server/bin/oozie-setup.sh
prepare-war -d /usr/hdp/current/oozie-server/libext-customer"
```

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

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

9. Make sure that following property is added in oozie-log4j.properties:

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

   where ${oozie.instance.id} is determined by oozie, automatically.

10. If you have custom Oozie actions, you must define them in 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>
    <property>
         <name>oozie.service.SparkConfigurationService.spark.configuration</
    name>
         <value>*=/etc/spark/conf/</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>
    ```

11. Configure HTTPS for the Oozie server.

    a. Create a self signed certificate or get certificate from a trusted CA for the Oozie Server

    b. Import the certificate to the client JDK trust store on all client nodes.

    c. In the Ambari Oozie configuration, set the following environment variables in `oozie-env.sh`, adding them if it does not exist:

       ```
       export OOZIE_HTTPS_PORT=11443
       export OOZIE_HTTPS_KEYSTORE_FILE=/home/oozie/.keystore
       export OOZIE_HTTPS_KEYSTORE_PASS=password
       ```

    d. Change `OOZIE_HTTP_PORT={{oozie_server_port}}` to `OOZIE_HTTP_PORT=11000`.

    e. Set the `oozie.base.url` to the HTTPS address.

    f. Save the configuration, and restart the Oozie components.

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

13. Check processes.

    ```
    ps -ef | grep -i oozie
    ```

14. When needed, the `/usr/hdp/current/oozie-server/bin/oozied.sh` script with the *`stop`* parameter stops the Oozie server.

3.17. Configure and Start Apache WebHCat

Note

The `su` commands in this section use "hdfs" to represent the HDFS Service user and webhcat to represent the WebHCat Service user. If you are using another name for these Service users, you need to substitute your Service user name for "hdfs" or "webhcat" in each of the `su` commands.

1. You must replace your configuration after upgrading. Copy `/etc/hive-webhcat/conf` from the template to the conf directory in webhcat hosts.

2. Modify the Apache WebHCat configuration files.

   a. Upload Pig, Hive and Sqoop tarballs to HDFS as the $HDFS_User (in this example, hdfs):

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0-<$version>/pig/pig.tar.gz /
hdp/apps/2.6.1.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0-<$version>/hive/hive.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0-<$version>/sqoop/sqoop.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0-<$version>/pig"

su - hdfs - "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0-<$version>/pig/pig.
tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0-<$version>/hive"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0-<$version>/hive/
hive.tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.6.1.0-<$version>/sqoop/
sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp"
```

   b. Add the following property if it is not present in webhcat-site.xml:

```
<property>
 <name>templeton.libjars</name>
```

```
 <value>/usr/hdp/current/zookeeper-client/zookeeper.jar,/usr/hdp/current/
hive-client/lib/hive-common.jar</value>
 <description>Jars to add the classpath.</description>
</property>
```

c. Add new proxy users, if needed. In core-site.xml, make sure the following properties are also set to allow WebHCat to impersonate your additional HDP 2.6.0 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 by 'hcat'.

`hadoop.proxyuser.hcat.hosts`

A comma-separated list of the hosts which are allowed to submit requests by 'hcat'.

3. Start WebHCat:

```
su - hcat -c "/usr/hdp/current/hive-webhcat/sbin/
webhcat_server.sh start"
```

4. Smoke test WebHCat.

a. If you have a non-secure cluster, on the WebHCat host machine, run the following command to check the status of WebHCat server:

```
curl http://$WEBHCAT_HOST_MACHINE:20111/templeton/v1/status
```

You should see the following return status:

```
"status":"ok","version":"v1"
```

b. If you are using a Kerberos secure cluster, run the following command:

```
curl --negotiate -u: http://$WEBHCAT_HOST_MACHINE:20111/
templeton/v1/status
```

You should see the following return status

```
{"status":"ok","version":"v1"}[machine@acme]$
```

3.18. Configure Apache Pig

Replace your configuration after upgrading. Copy /etc/pig/conf from the template to the conf directory in pig hosts.

3.19. Configure and Start Apache Sqoop

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

1. Replace your configuration after upgrading. Copy /etc/sqoop/conf from the template to the conf directory in sqoop hosts.

2. As the HDFS Service user, upload the Apache Sqoop tarball to HDFS.

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.6.1.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.6.1.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.6.1.0.0-<$version>/sqoop/sqoop.tar.
gz /hdp/apps/2.6.1.0.0-<$version>/sqoop/sqoop.tar.gz"

su - hdfs -c "hdfs dfs -chmod 444 /hdp/apps/2.6.1.0.0-<$version>/sqoop/
sqoop.tar.gz"
```

3. If you are using the MySQL database as a source or target, then the MySQL connector jar must be updated to 5.1.29 or later.

   Refer to the MySQL web site for information on updating the MySQL connector jar.

3.20. Configure, Start, and Validate Apache Flume

1. Replace your configuration after upgrading. Copy /etc/flume/conf from the template to the conf directory in Flume hosts.

2. By default, Apache Flume does not start running immediately upon installation.
   To validate your Flume upgrade, replace your default conf/flume.conf with the provided flume.conf file, restart Flume, and see if the data is flowing by examining the destination.

   Use this flume.conf file:

```
#1. Name the components on this agent
a1.sources = r1
a1.sinks = k1
a1.channels = c1
```

```
#2.Describe/configure the source
a1.sources.r1.type = seq

#3. Describe the sink
a1.sinks.k1.type = file_roll
a1.sinks.k1.channel = c1
a1.sinks.k1.sink.directory = /tmp/flume

#4. Use a channel which buffers events in memory
a1.channels.c1.type = memory

#5. Bind the source and sink to the channel
a1.sources.r1.channels = c1
a1.sinks.k1.channel = c1
```

3. After starting Flume, check /tmp/flume to see if there are any files there. The files should contain simple sequential numbers.
4. After validating, stop Flume and revert changes to flume.conf to prevent your disk from filling up.

3.21. Configure, Start, and Validate Apache Mahout

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

Replace your configuration after upgrading. Copy /etc/mahout/conf from the template to the conf directory in mahout hosts.

To validate Apache Mahout:

1. Create a test user named "testuser" in the Linux cluster and in HDFS, and log in as that user.
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 Linux server as `/tmp/sample-test.txt`.
4. Transfer the sample-test.txt file to a subdirectory of the testusers's HDFS home directory.

```
hdfs dfs -mkdir /user/testuser/testdata
hdfs dfs -put /tmp/sample-test.txt /user/testuser/testdata
```

5. Set up mahout to convert the plain text file sample-test.txt into a sequence file that is in the output directory mahouttest.

```
mahout seqdirectory --input /user/testuser/sample-test.txt --output /user/
testuser/mahouttest --charset utf-8
```

3.22. Configure and Start Hue

Note

The su commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the su commands.

For HDP 2.6.0, use the Hue version shipped with HDP 2.6.0. If you have a previous version of Hue, use the following steps to upgrade Hue.

1. Migrate hue.ini setting from your old hue.ini configuration file to new hue.ini configuration file.

2. If you are using the embedded SQLite database, remember to restore your database after upgrade.

   To restore the database from a backup, make sure the destination database is empty before copying (if necessary, rename or remove the current destination database), then copy your backup to the destination database. For example:

```
su - hue
cd /var/lib/hue
mv desktop.db desktop.db.old
sqlite3 desktop.db < ~/hue_backup/desktop.bak
exit
```

3. Synchronize Database

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

4. Start Hue. As a root user, run the following command on the Hue Server:

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

3.23. Configure and Start Apache Knox

When working with the Apache Knox Gateway in your Hadoop cluster, it is important you have the latest version of Knox installed so you can take advantage of new features and enhancements, in addition to ensuring your instance of Knox is in sync with other Hadoop components (e.g. Ranger, Spark, Hive, Hue, etc.) for stability and performance. For example, if you need to upgrade your Hadoop cluster from 2.2 to 2.3.x, you should

also make sure that your individual Hadoop components are also upgraded to the latest version.

HDP enables you to perform a rolling upgrade in 2.2.x and onward. A rolling upgrade means that you can upgrade a component, or the entire Hadoop stack, without losing service, and your users can continue to use the cluster and run jobs with no application or server downtime. The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL.

The main distinction between a rolling upgrade and a traditional upgrade implies the use of a Knox cluster for high availability capabilities. This means that you can require multiple instances of the gateway and a load balancer serving each gateway instance from a single URL. Once the upgrade process is completed, you are up and running with the latest version of Knox on each server you have designated as a Knox server.

Note

In this document, whenever you see a {} with a value inside, this denotes a value you must define.

3.23.1. Upgrade the Knox Gateway

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Knox manually to the latest version. Because “rolling upgrades” are now supported in HDP 2.6.0, it is not important which version of Knox you are currently running, only that you have an instance of the Knox Gateway running.

Note

If you have not already installed Knox, refer to the "Install the Knox RPMs on the Knox Server" section of the *Non-Ambari Cluster Installation Guide* for instructions on how to install and configure the Knox Gateway.

Before upgrading the Knox Gateway, there are a several steps you must follow to ensure your configuration files, settings, and topology files can be copied to the new Knox Gateway instance when the upgrade is complete, which are described below.

1. Back up your existing `conf` directory if you have not already done so.
2. Stop each Knox server if you have not already done so.

   ```
   su -l knox /usr/hdp/{the current Knox version}/knox/bin/gateway.sh stop
   ```

3. Select the HDP server version you are upgrading to after you have stopped each Knox server if you have not already done so.

   ```
   hdp-select set knox-server {the HDP server version}
   ```

4. 

   Note

   The `su` commands in this section use "knox" to represent the Knox Service user. If you are using another name for your Knox Service user, you need

to substitute your Knox Service user name for "knox" in each of the `su` commands.

For HDP 2.6.0, the default paths for Knox change. Upgrade Knox in order to update these paths.

a. Restore the backed up security directory. This places the master secret and keystores back in place for the new deployment.

b. Start the Gateway:

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

c. Unzip your previously saved configuration directory (the `conf` directory you backed up in step 1) into the new */var/log/knox/gateway.conf* directory to import these files.

d. Restart the Knox server to complete the upgrade.

```
su -l knox /usr/hdp/{the new HDP server version}/knox/bin/gateway.sh
 start
```

3.23.2. Verify the Knox Upgrade

To verify the upgrade was successful, follow the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful.

2. Verify you have cluster access using the `LISTSTATUS` WebHDFS API call.

```
curl -ivk -u {user}:{password} https://{knox host}:8443 /gateway/webhdfs/v1/
tmp?op=LISTSTATUS
```

3. Verify the Knox version using the Knox Admin service and Version API.

```
curl -ivk -u {adminuser}:{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

Note

The Admin API requires you to be a member of an Admin group, as specified in the `admin.xml` authorization provider.

When you have verified the Knox upgrade was successful, you can begin using Knox. If, however, the upgrade was unsuccessful, you need to downgrade the Knox Gateway to the previous version. The steps to downgrade the Knox Gateway are described in the next section.

3.23.3. Downgrade the Knox Gateway to the Previous Version

If the Knox Gateway upgrade was unsuccessful, you need to downgrade Knox to the previous version to ensure you have a working Knox Gateway for your cluster. To downgrade Knox, follow the steps listed below.

1. For each server running Knox, stop the server.

```
su -l knox /usr/hdp/{current HDP server version}/knox/bin/gateway.sh stop
```

2. Select the HDP server version you want to use to downgrade your Knox Gateway.

```
hdp-select set knox-server {previous HDP server version}
```

3. Restart the server.

```
su -l knox /usr/hdp/{previous HDP server version}/knox/bin/gateway.sh start
```

3.23.4. Verify the Knox Downgrade Was Successful

When the restart is complete, verify you are running an older version of Knox by following the steps listed below.

1. Navigate to the */var/log/knox/gateway* directory and check the `gateway.log`file for errors and an acknowledgement that the server restart was successful
2. Verify you have cluster access using the `LISTSTATUS`WebHDFS API call.
3. Check the Knox version using the Knox Admin service and Version API using the following command:

```
curl -ivk -u {adminuser}"{adminpassword} https://{knox host}:8443 /gateway/
admin/v1/version
```

3.24. Configure and Validate Apache Falcon

Note

In HDP 2.6.0, if authorization is enabled (for example, in the properties file with `*.falcon.security.authorization.enabled=true`") then Access Control List (ACL) is mandated for all entities.

Upgrade Apache Falcon after you have upgraded HDFS, Hive, Oozie, and Pig. Stop Oozie jobs while running Falcon.

1. Replace your configuration after upgrading. Copy /etc/falcon/conf from the template to the conf directory in falcon hosts.
2. Check your Falcon entities. There should be no changes, but in some cases you may need to update your entities post-upgrade.
3. In HDP 2.3.2 for Falcon, TLS is enabled by default. When TLS is enabled, Falcon starts on https://<falcon_host>.15443/. You can disable TLS by adding the following line to the startup.properties file.

   `"*.falcon.enableTLS=false`
4. If Transport Layer Security (TLS) is disabled, check the client.properties file to make sure the property "falcon.uri" is set as follows:

   `falcon.uri=http://<falcon_host>:15000/`

3.25. Configure and Start Apache Storm

Note

The su commands in this section use "zookeeper" to represent the ZooKeeper Service user. If you are using another name for your ZooKeeper Service user, you need to substitute your ZooKeeper Service user name for "zookeeper" in each of the su commands.

Apache Storm is fairly independent of changes to the HDP cluster, but you must upgrade Storm for rolling upgrade support in HDP 2.6.0 and be on the latest version of Storm.

1. Deactivate all running topologies.
2. Delete all states under zookeeper:

   ```
   /usr/hdp/current/zookeeper-client/bin/zkCli.sh (optionally in
   secure environment specify -server zk.server:port)

   rmr /storm
   ```

3. Delete all states under the storm-local directory:

   ```
   rm -rf <value of storm.local.dir>/*
   ```

4. Stop Storm Services on the storm node.
5. Stop ZooKeeper Services on the storm node.

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

6. Remove Storm and zookeeper from the storm node and install the HDP 2.6.0 version:
   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum erase storm

     yum erase zookeeper

     yum install storm

     yum install zookeeper
     ```

   - For **SLES**:

     ```
     zypper rm storm

     zypper rm zookeeper

     zypper install storm

     zypper install zookeeper
     ```

- For **Ubuntu/Debian**:

```
apt-get remove storm --purge

apt-get remove zookeeper --purge

apt-get install storm

apt-get install zookeeper
```

7. After upgrading Storm, replace your configuration in `/etc/storm/conf/storm.yaml` change references `backtype.storm` to `org.apache.storm` . This is needed because namespace migration in apache storm.

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

9. Ensure ZooKeeper is running. On the storm node, run the following command:

```
su - zookeeper -c "source /etc/zookeeper/conf/zookeeper-env.sh; export
 ZOOCFGDIR=/etc/zookeeper/conf; /usr/hdp/current/zookeeper-server/bin/
zkServer.sh start >> $ZOO_LOG_DIR/zoo.out\"
```

   where

   - $ZOO_LOG_DIR is the directory where ZooKeeper server logs are stored. For example, /var/log/zookeeper.

10. Start Storm processes, using a process controller (like supervisord) or manually. Start nimbus, then supervisor/ui/drpc/logviewer:

```
su - storm /usr/hdp/current/storm-nimbus/bin/storm nimbus
```

   You can use the same command syntax to start supervisor, ui, logviewer and drpc.

```
su - storm /usr/hdp/current/storm-supervisor/bin/storm
supervisor

su - storm /usr/hdp/current/storm-supervisor/bin/storm ui

su - storm /usr/hdp/current/storm-supervisor/bin/storm logviewer

su - storm /usr/hdp/current/storm-supervisor/bin/storm drpc
```

3.26. Configure and Start Apache Ranger

This section describes how to upgrade the Ranger service.

3.26.1. Preparing Your Cluster to Upgrade Ranger

If you are not currently using Ambari to manage your Hadoop cluster, you need to upgrade Apache Ranger manually to the latest version. This section describes the steps you need to follow to prepare your cluster for the Ranger upgrade.

1. Back up the following Ranger configuration directories(optional, since it has already been backed up in Getting ready to upgrade step):

   - Ranger Policy Administration Service

     ```
     /etc/ranger/admin/conf
     ```

   - Ranger UserSync

     ```
     /etc/ranger/usersync/conf
     ```

   - Ranger Plugins:

     - Hadoop

       ```
       /etc/hadoop/conf
       ```

     - Hive

       ```
       /etc/hive/conf
       ```

     - HBase

       ```
       /etc/hbase/conf
       ```

     - Knox

       ```
       /etc/knox/conf
       ```

     - Storm

       ```
       /etc/storm/conf
       ```

     - Kafka

       ```
       /etc/kafka/conf
       ```

2. Backup the Ranger Policy and Audit databases. Make sure to take note of the following details in the `install.properties` file:

   - db_host
   - db_name
   - db_user
   - db_password
   - policy manager configuration
   - LDAP directory configuration
   - LDAP settings
   - LDAP AD domain
   - LDAP URL

```
mysqldump -u root -p root ranger > dest_dir/filename.sql
mysqldump -u root -p root ranger_audit > dest_dir/audit_filename.sql
```

3.26.2. Stop the Ranger Services

Now that you have prepared your cluster for the Ranger upgrade, you need to stop the Ranger Admin and Ranger UserSync services. To stop the Ranger services, perform the steps described below.

1. Stop the Ranger Policy Admin service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-admin stop
   ```

2. Stop the Ranger UserSync service. When the service is stopped, you receive an acknowledgement from the server that the service has been stopped.

   ```
   service ranger-usersync stop
   ```

3. Stop the applicable services using the Ranger plugin (HDFS, HBase, Hive, Knox, Storm, YARN, Kafka).

   See Stopping HDP Services for more information.

3.26.3. Preparing the Cluster for Upgrade

Before you begin the upgrade process, you need to perform a series of steps to prepare the cluster for upgrade. These steps are described in the "*Getting Ready To Upgrade*" section of this guide, which you need to follow before continuing to upgrade Ranger. Some of these steps include:

- Backing up HDP directories
- Stopping all long-running applications and services.
- Backing up the Hive and Oozie metastore databases.
- Backing up Hue
- Backing up specific directories and configurations

3.26.4. Registering the HDP 2.6.0 Repo

After you have prepared your cluster for the upgrade, you need to register the HDP 2.6.0 repo. This requires you to perform the following steps:

1. (Optional) The Ranger components should already have installed in the at the beginning of the HDP upgrade process, but you can use the following commands to confirm that the Ranger packages have been installed:

   ```
   hdp-select status ranger-admin
   hdp-select status ranger-usersync
   ```

If the packages have not been installed, you can use the install commands specific to your OS. For example, for RHEL/CentOS you would use the following commands to install the packages.

```
yum install ranger_2_5_*-admin
yum install ranger_2_5_*-usersync
```

2. Select the Ranger Admin and Ranger UserSync versions you want to use.

```
hdp-select set ranger-admin <HDP_server_version>
hdp-select set ranger-usersync <HDP_server_version>
```

3. Change ownership of `/etc/ranger/admin/conf/` and `/etc/ranger/usersync/conf/` to `ranger:ranger`:

```
chown -R ranger:ranger /etc/ranger/admin/conf/
```

4. Solr must be installed and configured before installing RangerAdmin or any of the Ranger component plugins.

   For information regarding installation and configuration of Solr, see Using Apache Solr for Ranger Audits.

5. Update the `install.properties` file to migrate the database credentials properties and POLICYMGR_EXTERNAL-URL property from HDP 2.3. to HDP 2.6.0.

Table 3.4. Ranger_Admin install.properties names and values

Property Name	Property Value
DB_FLAVOR	MySql(|ORACLE|POSTGRES|MSSQL|SQLA)
db_root_user	root
db_root_password	Password of db (eg: vagrant)
db_host	Hostname : where your db does exist
policymgr_external_url	http://<hostname>:6080
policymgr_http_enabled	true
authentication_method	UNIX(LDAP|ACTIVE_DIRECTORY|UNIX|NONE)
audit_store	solr
audit_solr_urls	http://<solr_host>:6083/solr/ranger_audits

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

6. Install the Ranger Admin component. Be sure to set the JAVA_HOME environment variable if it is not already set.

```
cd /usr/hdp/current/ranger-admin/

cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-site.xml ews/webapp/
WEB-INF/classes/conf/
```

```
cp  ews/webapp/WEB-INF/classes/conf.dist/ranger-admin-default-site.xml ews/
webapp/WEB-INF/classes/conf/

cp  ews/webapp/WEB-INF/classes/conf.dist/security-applicationContext.xml
 ews/webapp/WEB-INF/classes/conf/

./setup.sh
```

This should successfully install the Ranger Admin component.

7. Start the Ranger Admin component.

```
service ranger-admin start
```

8. Configure and setup the Ranger UserSync component by migrating the properties from the HDP 2.3 `install.properties` file (POLICY_MGR_URL, SYNC_SOURCE and LDAP/AD properties.

Table 3.5. Ranger_Usersync install.properties names and values

Property Name	Property Value
POLICY_MGR_URL	http://<hostname>:6080
SYNC_SOURCE	unix
SYNC_INTERVAL	5

Note

When you migrate to new version, you have to remove `/user/bin/ranger-admin`, which points to the older version of the ranger start file, `/usr/hdp/<version>/ranger-admin/ews/ranger-admin-services.sh`. After you remove this file, you have to run setup again.

9. Install the Ranger UserSync component. Be sure to set the JAVA_HOME component if it is not already set.

```
cd /usr/hdp/current/ranger-usersync/

./setup.sh
```

10. Start the Ranger UserSync component.

```
service ranger-usersync start
```

3.26.5. Install the Ranger Components

Next, you need to re-install each Ranger component again to ensure you have the latest version. Because you have already upgraded your HDP stack, you only need to follow the instructions in the *Non-Ambari Cluster Installation Guide* to install each Ranger component. You must install the following Ranger components:

- Ranger Policy Admin
- Ranger UserSync
- Ranger Plugins:

- HDFS
- HBase
- Hive
- Knox
- Storm
- YARN
- Kafka
- Solr

Note

When installing each Ranger component, you also need to make sure you upgrade each individual component to version 2.6.0 before restarting each service.

3.26.6. Restart the Ranger Services

Once you have re-installed each Ranger component, you then need to restart these components to ensure the new configurations are loaded in your cluster. This includes restarting the Policy Admin and UserSync components, NameNode, and each Ranger plugin.

Note

Before restarting the NameNode, make sure to remove the `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable the NameNode after finishing the upgrade.

The *Non-Ambari Cluster Installation Guide* describes how you can start the following Ranger services:

- Ranger Policy Admin service

```
service ranger-admin start
```

- Ranger UserSync service

```
service ranger-usersync start
```

3.26.7. Enable Ranger Plugins

The final step in the Ranger upgrade process requires you to re-enable the Ranger plugins. Although you are only required to enable HDFS in your cluster, you should re-enable all of the Ranger plugins because class names have changed for the 2.6.0 release, and to ensure smooth operation of Ranger services in your cluster.

Note

When you enable each Ranger plugin, be sure to remove all 2.3 class name values.

Note

Re-enabling a Ranger plugin does not affect policies you have already created. As long as you use the same database as the Policy store, all of your data remains intact.

To re-enable the Ranger plugins, use the links listed below to access instructions in the *Non-Ambari Cluster Installation* guide that describe editing the `install.properties` file and enabling the Ranger plugins:

Important

Before enabling the HDFS plugin, remove `set-hdfs-plugin-env.sh` from `/etc/hadoop/conf`. You need to re-enable this plugin after the upgrade is complete.

- HDFS Plugin
- YARN Plugin
- Kafka Plugin
- HBase Plugin
- Hive Plugin
- Knox Plugin
- Storm Plugin

3.26.8. Enable KMS Configuration

Configure and setup the Ranger KMS configuration by editing the KMS `install.properties` file:

Table 3.6. KMS install.properties names and values

Property Name	Property Name
DB_FLAVOR	MYSQL
db_root_user	foot
db_root_password	<db password>
db_host	<db hostname>
db_name	rangerkms (default)
db_user	rangerkms (default)
db_password	<kms db password>
hadoop_conf	/etc/hadoop/conf
POLICY_MGR_URL	http://<hostname>:6080

Property Name	Property Name
REPOSITORY_NAME	kmsdev
XAAUDIT.SOLR.ENABLE	false (default), change to true if desired
XAAUDIT.SOLR.URL	http://<solr_host>:6083/solr/ranger_audits

3.26.9. Configure and Start Apache Ranger on a Kerberized Cluster

Beginning with HDP 2.6.0, kerberos authentication is supported for Ranger and its plugins. Use the this section if you have an HDP 2.3 cluster in a kerberized environment with Ranger in simple authentication mode that you want to upgrade.

For additional information regarding Kerberos, refer to Kerberos Overview in the *Hadoop Security Guide*.

3.26.9.1. Create Keytabs and Principals

1. Follow these steps to create keytabs and principals:

   For Ranger Admin:

   a. Create rangeradmin/<FQDN of Ranger Admin>@<REALM>.

   b.
   ```
   > kadmin.local
   > addprinc -randkey rangeradmin/<FQDN of Ranger Admin>
   Eg: addprinc -randkey rangeradmin/ranger-upgrade-0707-2.openstacklocal
   > xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/<FQDN of
    Ranger Admin>@<REALM>
   Eg: xst -k /etc/security/keytabs/rangeradmin.keytab rangeradmin/ranger-
   upgrade-0707-2.openstacklocal/EXAMPLE.COM
   >exit
   ```

   For Ranger Lookup:

   a. Create rangerlookup/<FQDN of Ranger Admin>@<REALM>.

   b.
   ```
   > kadmin.local
   > addprinc -randkey  rangerlookup/<FQDN of Ranger Admin>
   Eg: addprinc -randkey rangerlookup/ranger-upgrade-0707-2.openstacklocal
   > xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/<FQDN of
    Ranger  Admin>@<REALM>
   Eg: xst -k /etc/security/keytabs/rangerlookup.keytab rangerlookup/ranger-
   upgrade-0707-2.openstacklocal@EXAMPLE.COM
   > exit
   ```

   For Ranger Usersync:

   a. Create rangerusersync/<FQDN>@<REALM>.

   b.
   ```
   > kadmin.local
   > addprinc -randkey rangerusersync/<FQDN of Ranger usersync>
   Eg: addprinc -randkey rangerusersync/ranger-upgrade-0707-2.openstacklocal
   ```

```
> xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerusersync.keytab rangerusersync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger Tagsync

a. Create rangertagsync/<FQDN>@<REALM>.

b.
```
> kadmin.local
> addprinc -randkey rangertagsync/<FQDN of Ranger tagsync>
Eg: addprinc -randkey rangertagsync/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
<FQDN>@<REALM>
Eg: xst -k /etc/security/keytabs/rangertagsync.keytab rangertagsync/
ranger-upgrade-0707-2.openstacklocal@EXAMPLE.COM
> exit
```

For Ranger KMS:

a. Create rangerkms/<FQDN of Ranger Admin>@<REALM>

b.
```
> kadmin.local
> addprinc -randkey rangerkms/<FQDN of Ranger Admin>
Eg: addprinc -randkey rangerkms/ranger-upgrade-0707-2.openstacklocal
> xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/<FQDN of Ranger
 Admin>@<REALM>
Eg: xst -k /etc/security/keytabs/rangerkms.keytab rangerkms/ranger-
upgrade-0707-2.openstacklocal/EXAMPLE.COM
> exit
```

2. If the Kerberos server and admin are on different hosts, copy the keytab on the admin host, assign permission to user `ranger`, and change the permissions.

```
scp the <rangeradmin_keytab_file> to <new_path>
chown ranger <rangeradmin_keytab_path>
chmod 400 <rangeradmin_keytab_path>
```

3. Use **kdestroy** to delete the Keberos credentials cache file.

4. Set the following properties and values in the `/etc/hadoop/conf/core-site.xml` file:

Table 3.7. Properties for the /etc/hadoop/conf/core-site.xml file

Property Name	Property Value
fs.defaultFS	hdfs://ranger-upgrade-0707-2.openstacklocal:8020
hadoop.security.authentication	kerberos
hadoop.security.authorization	true
hadoop.security.auth_to_local	RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/ RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*// RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/ RULE:[2:$1@$0]

Property Name	Property Value
	(rangeradmin@EXAMPLE.COM)s/.*/ranger/ RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/ DEFAULT

The following is an example of a `core-site.xml` file with the properties set for Kerberos:

```
<configuration>

    <property>
      <name>fs.defaultFS</name>
      <value>hdfs://ranger-upgrade-0707-2.openstacklocal:8020</value>
      <final>true</final>
    </property>

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

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

<property>
      <name>hadoop.security.auth_to_local</name>
      <value>RULE:[1:$1@$0](ambari-qa-cluster1@EXAMPLE.COM)s/.*/ambari-qa/
RULE:[1:$1@$0](.*@EXAMPLE.COM)s/@.*//
RULE:[2:$1@$0](dn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](nn@EXAMPLE.COM)s/.*/hdfs/
RULE:[2:$1@$0](rangeradmin@EXAMPLE.COM)s/.*/ranger/
RULE:[2:$1@$0](rangerusersync@EXAMPLE.COM)s/.*/rangerusersync/
DEFAULT</value>
    </property>

</configuration>
```

3.26.9.2. Run Setup Again for Ranger Admin

1. Stop the ranger-admin service.
2. Add the following properties and values to the ranger-admin `install.properties` file:

Table 3.8. Ranger-admin install.properties

Property Name	Property Value
spnego_principal	HTTP/<FQDN_OF_Ranger_Admin_Cluster>@<REALM>
spnego_keytab	<HTTP keytab path>
token_valid	30
cookie_domain	<FQDN_OF_Ranger_Admin_Cluster>
cookie_path	/
admin_principal	rangeradmin/<FQDN_OF_Ranger_Admin_Cluster>@<REALM>

Property Name	Property Value
admin_keytab	<rangeradmin keytab path>
lookup_principal	rangerlookup/<FQDN_OF_Ranger_Admin_Cluster>@<REALM>
lookup_keytab	<rangerlookup keytab path>
hadoop_conf	/etc/hadoop/conf

3. Execute the `setup.sh` script.
4. Start the ranger-admin service.
5. Stop the ranger-usersync service.
6. Add the following properties and values to the ranger-sync `install.properties` file:

Table 3.9.

Property Name	Property Value
usersync_principal	rangerusersync/<FQDN>@<REALM>
usersyn_keytab	<rangerusersync keytab path>
hadoop_conf	/etc/hadoop/conf

7. Execute the `setup.sh` script.
8. Start the ranger-sync service.
9. Stop the ranger-tagsync service.
10. Add the following properties and values to the ranger-tagsync `install.properties` file:

Table 3.10. Ranger-tagsync install.properties and values

Property Name	Property Value
tagsync_principal	rangertagsync/<FQDN>@<REALM>
tagsync_keytab	<rangertagsync keytab path>
hadoop_conf	/etc/hadoop/conf

11. Execute the `setup.sh` script.
12. Start the ranger-tagsync service.
13. Stop the ranger-kms service.
14. Add the following properties and values to the ranger-kms `install.properties` file:

Table 3.11. Ranger-kms install.properties and values

Property Name	Property Value
kms_principal	rangerkms/<FQDN>@<REALM>
kms_keytab	<rangerkms keytab path>
hadoop_conf	/etc/hadoop/conf

3.26.9.3. Install and Enable the Ranger HDFS Plugin

This section documents how to install and enable the Ranger HDFS plugin. You might want to consider making similar changes for the other Ranger plugins that you are using.

The Ranger HDFS plugin is located at `/usr/hdp/<version>/ranger-hdfs-plugin`.

Follow these steps to install and enable the Ranger HDFS Plugin:

1. Edit the relevant lines in the `install.properties` file:

```
POLICY_MGR_URL=http://<FQDN of ranger admin host>:6080
REPOSITORY_NAME=hadoopdev
Audit info (Solr/HDFS options available)
```

2. Enter the following commands to enable the HDFS plugin:

```
export JAVA_HOME=<JAVA Path>
./enable-hdfs-plugin.sh
```

3. Start and stop the namenode:

```
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"
```

4. In the custom repo configurationfile, add the component user, for example, `hdfs`, as a value for each of the following properties:
   - policy.download.auth.users or policy.grantrevoke.auth.users
   - tag.download.auth.users
5. Verify that the plugin communicates with Ranger admin using the **Audit # plugins** tab.
6. Set the following properties in the `hdfs-site.xml` file:

Table 3.12. hdfs-site.xml Property Names and Values

Property Name	Property Value
dfs.permissions.enabled	true
dfs.permissions.supergroup	hdfs
dfs.block.access.token.enable	true
dfs.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.secondary.namenode.kerberos.principal	nn/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos.principal	HTTP/_HOST@EXAMPLE.COM
dfs.web.authentication.kerberos. keytab	/etc/security/keytabs/spnego.service.keytab
dfs.datanode.kerberos.principal	dn/_HOST@EXAMPLE.COM
dfs.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.secondary.namenode.keytab.file	/etc/security/keytabs/nn.service.keytab
dfs.datanode.keytab.file	/etc/security/keytabs/dn.service.keytab
dfs.https.port	50470
dfs.namenode.https-address	Example:ip-10-111-59-170.ec2.internal:50470
dfs.datanode.data.dir.perm	750

Property Name	Property Value
dfs.cluster.administrators	hdfs
dfs.namenode.kerberos.internal. spnego.principal	${dfs.web.authentication.kerberos.principal}
dfs.secondary.namenode.kerberos. internal.spnego.principal	${dfs.web.authentication.kerberos.principal}

The following is an example of a `hdfs-site.xml` file with the properties set for Kerberos:

```
<property>
     <name>dfs.permissions</name>
     <value>true</value>
     <description> If "true", enable permission checking in
     HDFS. If "false", permission checking is turned
     off, but all other behavior is
     unchanged. Switching from one parameter value to the other does
     not change the mode, owner or group of files or
     directories. </description>
</property>

<property>
     <name>dfs.permissions.supergroup</name>
     <value>hdfs</value>
     <description>The name of the group of
     super-users.</description>
</property>

<property>
     <name>dfs.namenode.handler.count</name>
     <value>100</value>
     <description>Added to grow Queue size so that more
     client connections are allowed</description>
</property>

<property>
     <name>ipc.server.max.response.size</name>
     <value>5242880</value>
</property>

<property>
     <name>dfs.block.access.token.enable</name>
     <value>true</value>
     <description> If "true", access tokens are used as capabilities
     for accessing datanodes. If "false", no access tokens are checked on
     accessing datanodes. </description>
</property>

<property>
     <name>dfs.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description> Kerberos principal name for the
     NameNode </description>
</property>

<property>
     <name>dfs.secondary.namenode.kerberos.principal</name>
     <value>nn/_HOST@EXAMPLE.COM</value>
     <description>Kerberos principal name for the secondary NameNode.
```

```
     </description>
</property>

<property>
     <!--cluster variant -->
     <name>dfs.secondary.http.address</name>
     <value>ip-10-72-235-178.ec2.internal:50090</value>
     <description>Address of secondary namenode web server</description>
</property>

<property>
     <name>dfs.secondary.https.port</name>
     <value>50490</value>
     <description>The https port where secondary-namenode
     binds</description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.principal</name>
     <value>HTTP/_HOST@EXAMPLE.COM</value>
     <description> The HTTP Kerberos principal used by Hadoop-Auth in the
 HTTP endpoint.
     The HTTP Kerberos principal MUST start with 'HTTP/' per Kerberos HTTP
     SPNEGO specification.
     </description>
</property>

<property>
     <name>dfs.web.authentication.kerberos.keytab</name>
     <value>/etc/security/keytabs/spnego.service.keytab</value>
     <description>The Kerberos keytab file with the credentials for the HTTP
     Kerberos principal used by Hadoop-Auth in the HTTP endpoint.
     </description>
</property>

<property>
     <name>dfs.datanode.kerberos.principal</name>
     <value>dn/_HOST@EXAMPLE.COM</value>
     <description>
     The Kerberos principal that the DataNode runs as. "_HOST" is replaced
 by the real
     host name.
     </description>
</property>

<property>
     <name>dfs.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
     </description>
</property>

<property>
     <name>dfs.secondary.namenode.keytab.file</name>
     <value>/etc/security/keytabs/nn.service.keytab</value>
     <description>
     Combined keytab file containing the namenode service and host
     principals.
```

```
     </description>
</property>

<property>
     <name>dfs.datanode.keytab.file</name>
     <value>/etc/security/keytabs/dn.service.keytab</value>
     <description>
     The filename of the keytab file for the DataNode.
     </description>
</property>

<property>
     <name>dfs.https.port</name>
     <value>50470</value>
     <description>The https port where namenode
     binds</description>
</property>

<property>
     <name>dfs.https.address</name>
     <value>ip-10-111-59-170.ec2.internal:50470</value>
     <description>The https address where namenode binds</description>
</property>

<property>
     <name>dfs.datanode.data.dir.perm</name>
     <value>750</value>
     <description>The permissions that should be there on
     dfs.data.dir directories. The datanode will not come up if the
     permissions are different on existing dfs.data.dir directories. If
     the directories don't exist, they will be created with this
     permission.</description>
</property>

<property>
     <name>dfs.access.time.precision</name>
     <value>0</value>
     <description>The access time for HDFS file is precise upto this
     value.The default value is 1 hour. Setting a value of 0
     disables access times for HDFS.
     </description>
</property>

<property>
     <name>dfs.cluster.administrators</name>
     <value> hdfs</value>
     <description>ACL for who all can view the default
     servlets in the HDFS</description>
</property>

<property>
     <name>ipc.server.read.threadpool.size</name>
     <value>5</value>
     <description></description>
</property>

<property>
     <name>dfs.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

```
<property>
     <name>dfs.secondary.namenode.kerberos.internal.spnego.principal</name>
     <value>${dfs.web.authentication.kerberos.principal}</value>
</property>
```

7. For Download Policy to be successful, use the Ranger UI to update the service configuration with the following custom properties for each supported component:

```
policy.download.auth.users=<Component service user>
tag.download.auth.users=<Component service user>(if tag download)
```

8. For Grant/Revoke for Hive and Hbase to be successful, use the Ranger UI to update the service configuration with the following custom property:

```
policy.grantrevoke.auth.users = <Component service user>
```

9. For Test Connection and Resource Lookup to be successful, use the Ranger UI to add lookup user in the permission list of the policies.

3.27. Configuring and Upgrading Apache Spark

Before you can upgrade Apache Spark, you must have first upgraded your HDP components to the latest version (in this case, 2.6.0). This section assumes that you have already upgraded your components for HDP 2.6.0. If you have not already completed these steps, return to Getting Ready to Upgrade and Upgrade 2.3 Components for instructions on how to upgrade your HDP components to 2.6.0.

The upgrade process installs Spark version 1.6. If you want to use Spark version 2, install version 2 after finishing the HDP 2.6 upgrade process. For more information, see Installing and Configuring Apache Spark 2 in the *Command Line Installation Guide*.

To upgrade Spark, start the service and update configurations.

1. Stop the Spark `history-server`. If you are using the Spark `thrift-server`, stop the `thrift-server`.

```
su - spark -c "$SPARK_HOME/sbin/stop-history-server.sh"
su - spark -c "$SPARK_HOME/sbin/stop-thriftserver.sh"
```

2. Remove any reference to `hdp.version` from the Spark configuration files.

   Remove `spark.yarn.services` property from `spark-defaults.conf`.

   Make sure that `spark.history.provider`, if present, is set to `org.apache.spark.deploy.history.FsHistoryProvider` (the default).

3. Restart the `history-server`:

```
su - spark -c "$SPARK_HOME/sbin/start-history-server.sh"
```

4. If you are using the Spark thrift-server, restart the `thrift-server`. See (Optional) Starting the Spark Thrift Server.

5. Validate the Spark installation. As user `spark`, run the examples in Running Spark Applications in the *Spark Guide*.

For additional configuration information, see the Spark Guide.

3.28. Upgrade Apache Slider

To upgrade Apache Slider, simply upgrade the Slider client.

1. Upgrade Slider client:

```
hdp-select set slider-client 2.6.1.0.0-<version>
slider version
```

3.29. Upgrade Apache Kafka

Upgrade each Apache Kafka node one at a time.

You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

You can stop one Kafka broker at a time, upgrade the Kafka binaries to HDP 2.6, and start the broker, before stopping the next broker.

Note

If you are willing to accept downtime, you can take all of the brokers down, update the code, and restart all of the brokers. By default, the brokers start with the new protocol.

You can bump the protocol version and restart, at any time after the brokers are upgraded.

The following steps are specific to upgrading to HDP 2.6 from a version lower than HDP 2.5. Follow these steps prior to upgrading Kafka:

1. Add the following properties to the `server.properties` file on all brokers:

```
inter.broker.protocol.version=CURRENT_KAFKA_VERSION
log.message.format.version=CURRENT_KAFKA_VERSION
```

   Examples of CURRENT_KAFKA_VERSION are `0.8.2.0`, `0.9.0.0` or `0.10.0.0`.

2. Upgrade the brokers one at a time: shut down the broker, update the code, and restart it.

3. Once the entire cluster has been upgraded, bump the protocol version by setting inter.broker.protocol.version to `0.10.1.0`.

4. If your previous message format is `0.10.0`, change log.message.format.version to `0.10.1` (this is a no-op as the message format is the same for both 0.10.0 and 0.10.1). If your previous message format version is lower than `0.10.0`, do not change log.message.format.version yet - this parameter should only change once all consumers have been upgraded to `0.10.0.0` or later.

5. Restart the brokers one at a time for the new protocol version to take effect.

6. If log.message.format.version is lower than `0.10.0`, wait until all consumers have been upgraded to `0.10.0` or later. Once all consumers are upgraded to `0.10.0`, change log.message.format.version to `0.10.1` on each broker, and restart each broker one at a time.

Follow these steps to upgrade Kafka:

1. Shut down the current Kafka daemon, switch to the new version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.3.4.0-2633
su - kafka -c "usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

3. If the upgrade process fails, follow the steps in "Downgrading Kafka" to return to your previous version of Kafka.

3.29.1. Downgrading Kafka

Downgrade each Kafka node one at a time. You can stop each Kafka broker and upgrade the component without downtime if you have enough replication for your topic.

1. Shut down the current Kafka daemon, switch to the previous version, and start the daemon:

```
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka stop"
hdp-select set kafka-broker 2.6.1.0-2041
su - kafka -c "/usr/hdp/current/kafka-broker/bin/kafka start"
```

2. To verify that the Kafka daemon joined the cluster, create a topic and submit it to Kafka. Send a test message for that topic, and then validate that it was received by a consumer.

3.30. Finalize the Upgrade

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hdfs" is used to represent the HDFS Service user. If you are using another name for your Service users, you need to substitute your Service user name in each of the `su` commands.

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

1. Verify your file system health before finalizing the upgrade. After you finalize an upgrade, your backup is discarded!

2. As the $HDFS_USER, enter:

```
su - hdfs -c "dfsadmin -finalizeUpgrade"
```

3.31. Migrate the Audit Logs from DB to Solr

Beginning with HDP 2.6.0, audit log to DB support has been removed. If your logs were previously stored on DB, you can migrate the logs to Solr. Refer to Migrating Audit Logs from DB to Solr.

3.32. Install New HDP 2.6.0 Services

1. Install new HDP 2.6.0 Services (see the Non-Ambari Cluster Installation Guide):

   - SmartSense: a next generation subscription model that features upgrade and configuration recommendations.