Hortonworks Data Platform

Upgrading HDP Manually

(July 21, 2015)

docs.cloudera.com

Hortonworks Data Platform: Upgrading HDP Manually

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache Hadoop projects including MapReduce, Hadoop Distributed File System (HDFS), HCatalog, Pig, Hive, HBase, ZooKeeper and Ambari. Hortonworks is the major contributor of code and patches to many of these projects. These projects have been integrated and tested as part of the Hortonworks Data Platform release process and installation and configuration tools have also been included.

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

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

Table of Contents

List of Tables

1. Upgrade from HDP 2.2 to HDP 2.3 Manually

Important

If you installed and manage HDP 2.2 with Ambari, **you must use the Ambari Upgrade Guide** to perform the the HDP 2.2 to HDP 2.3.0 upgrade.

Note

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.

Starting with HDP 2.2, HDP 2.3.0 supports side-by-side installation of HDP 2.2 and subsequent releases, which lets you perform rolling upgrades on your cluster and improve execution times for in-place upgrade. To support side-by-side installation, the HDP package version naming convention for both RPMs and Debs has changed to include the HDP product version. For example, hadoop-hdfs is now hadoop-2.3.0-hdfs. HDP 2.2 marked the first release where HDP rpms, debs, and directories contained versions in the names to permit side-by-side installations of later HDP releases. To select from the releases you have installed side-by-side, Hortonworks provided hdp-select, a command that lets you select the active version of HDP from the versions you have installed.

Now with HDP 2.3.0, you can take advantage of this versioned installation to perform Rolling Upgrade from 2.2 to 2.3.0. However, 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.

This chapter provides instructions on how to manually upgrade to HDP 2.3.0 from the HDP 2.2 release. It assumes the existing HDP 2.2 was also installed manually.

The HDP packages for a complete installation of HDP 2.3.0 will occupy about 2.5 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 2.3.0 from HDP 2.2:

1. Get Ready to Upgrade

2. Upgrade HDP 2.2 components

3. Use hdp-select to symlink the HDP 2.3.0 components into "current," in place of the former HDP 2.2 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 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 2.3.0 services

1.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.2 to HDP 2.3.0 versions and adding the new HDP 2.3.0 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 2.3.0 will take up about 2.5 GB of disk space.

The first step is to ensure you keep a backup copy of your HDP 2.2 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 will 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/phoenix/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/falcon/conf
- /etc/slider/conf/
- /etc/storm/conf/
- /etc/storm-slider-client/conf/
- /etc/spark/conf/
- /etc/ranger/admin/conf, /etc/ranger/usersync/conf (if Ranger is installed)
- 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

Component	Command
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/current/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`

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 HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec && /usr/hdp/current/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/hdp/current/hadoop-yarn-timelineserver/sbin/yarn-daemon.sh`

Component	Command
	`--config /etc/hadoop/conf stop timelineserver"`

Storm	Deactivate all running topologies:
	`storm kill topology-name`
	Delete all states under zookeeper:
	`/usr/hdp/current/zookeeper-client/ bin/zkCli.sh (optionally in secure environment specify -server zk.server:port)`
	`rmr /storm`
	Delete all states under the storm-local directory:
	`rm -rf <value of stormlocal.dir>`
	Stop Storm Serves on the storm node:
	`sudo service supervisord stop`
	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"`

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 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 1.2. Oozie Metastore Database Backup and Restore

Database Type	Backup	Restore
MySQL	mysqldump $dbname > $outputfilename.sql For example: mysqldump oozie > /tmp/mydir/backup_oozie.sql	mysql $dbname < $inputfilename.sql For example: mysql oozie < /tmp/mydir/backup_oozie.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql For example: `sudo -u postgres pg_dump oozie > /tmp/mydir/backup_oozie.sql`	sudo -u $username psql $databasename < $inputfilename.sql For example: `sudo -u postgres psql oozie < /tmp/mydir/backup_oozie.sql`
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: 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	/etc/init.d/hue stop

Database Type	Backup	Restore
	su $HUE_USER mkdir ~/hue_backup sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak /etc/init.d/hue start	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 -c "hdfs dfsadmin -report > dfs-old-report-l.log"
```

c. **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.

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

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:
	
	`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-client/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:

Component	Command
	`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.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 the edit logs in `${dfs.namenode.name.dir}/current/edits*` are empty.

a. Run this command on the Active Namenode machine:

```
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 will print 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 will then suffix 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` will rename 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 before upgrading HDP. If you are running JDK 1.7, no action is required.

Important

If you want to upgrade to from JDK 1.7 to JDK 1.8, you must update the HDP stack before upgrading to JDK 1.8. For example, the high-level process should follow:

a. Run HDP 2.2 with JDK 1.7.

b. Perform the stack upgrade to HDP 2.3.0. See Upgrade HDP 2.2 Components.

c. Upgrade JDK from 1.7 to 1.8.

1.2. Upgrade HDP 2.2 Components

The upgrade process to HDP 2.3.0 involves the following steps. See the Release Notes for the HDP 2.3.0.0 repo information. Select your OS:

RHEL/CentOS/Oracle 6

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.2 components. This command uninstalls the HDP 2.2 components. It leaves the user data, and metadata, but removes your configurations. Refer to the HDP Component Files for RHEL/CentOS/Oracle6 table for the names of the files that need to be removed for each component and use the following format:

   ```
   yum erase "accumulo_$VERSION_$BUILD*" "atlas_metadata_$VERSION_$BUILD*"
    "datafu_$VERSION_$BUILD*" "falcon_$VERSION_$BUILD*" "flume_$VERSION_
   $BUILD*" "hadoop_$VERSION_$BUILD*" "hadoop_$VERSION_$BUILD-client*" "hadoop_
   $VERSION_$BUILD-hdfs*" "hadooplzo_$VERSION_$BUILD*" ...
   ```

   Where:

   - $VERSION is the HDP version number in the following format: 2_3_0_0
   - $HUE-VERSION is the Hue version number in the following format : 2.6.1.2.3.0.0
   - $BUILD is the HDP build number in the following format : 3518

   So, the file name will look like: accumulo-2_3_0_0_3518*

Table 1.4. HDP Component Files for RHEL/CentOS/Oracle6

Component	Associated Files
Accumulo	accumulo_$VERSION_$BUILD*
Atlas	atlas_metadata_$VERSION_$BUILD*
Datafu	datafu_$VERSION_$BUILD*
Falcon	falcon_$VERSION_$BUILD*
Flume	flume_$VERSION_$BUILD*
Hadoop (includes hadoop_client and hadoop_hdfs)	hadoop_$VERSION_$BUILD*

Component	Associated Files
	hadoop_$VERSION_$BUILD-client* hadoop_$VERSION_$BUILD-hdfs* hadooplzo_$VERSION_$BUILD*
HBase	hbase_$VERSION_$BUILD*
Hive	hive_$VERSION_$BUILD* hive_hbase_$VERSION_$BUILD* hive_hcatalog_$VERSION_$BUILD* hive_jdbc_$VERSION_$BUILD* hive_metastore_$VERSION_$BUILD* hive_server_$VERSION_$BUILD* hive_webhcathive_hbase_$VERSION_$BUILD* ranger_hive_$VERSION_$BUILD*
Hue	hue_$HUE_VERSION_$BUILD* hue_beeswax_$HUE_VERSION_$BUILD* hue_common_$HUE_VERSION_$BUILD* hue_hcatalog_$HUE_VERSION_$BUILD* hue_oozie_$HUE_VERSION_$BUILD* hue_pig_$HUE_VERSION_$BUILD* hue_server_$HUE_VERSION_$BUILD*
Kafka	kafka_$VERSION_$BUILD*
Knox	knox_$VERSION_$BUILD*
Mahout	mahout_$VERSION_$BUILD*
MapReduce	hadoop_$VERSION_$BUILD-mapreduce*
Oozie	oozie_$VERSION_$BUILD*
Phoenix	phoenix_$VERSION_$BUILD
Pig	ranger_$VERSION_$BUILD*
Ranger	ranger_$VERSION_$BUILD*
Slider	slider_$VERSION_$BUILD*
Spark	spark_$VERSION_$BUILD*
Sqoop	sqoop_$VERSION_$BUILD*
Storm	storm_$VERSION_$BUILD*
Tez	tez_$VERSION_$BUILD*
YARN	hadoop_$VERSION_$BUILD-yarn*
ZooKeeper	zookeeper_$VERSION_$BUILD*

3. Validate that all HDP 2.2 component binaries are uninstalled:

```
yum list installed | grep @HDP2.2
```

4. Remove your old hdp.repo file:

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

5. Install the HDP 2.3.0 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.3.
     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-2.3.0 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-2.3.0 Hortonworks Data Platform Version - HDP-2.3.0
     ```

6. Install the HDP 2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.2 components:

   ```
   yum install "hadoop" "hadoop-hdfs" "hadoop-libhdfs" "hadoop-yarn" "hadoop-
   mapreduce" "hadoop-client" "hadoop-lzo" "openssl" "hive-webhcat" "hive-
   hcatalog" "oozie" "collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase"
    "hue" "hive" "tez" "storm" "falcon" "flume" "phoenix" "accumulo" "mahout"
    "knox" "kafka" "spark" "ranger" "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 SP 1

1. On all hosts, clean the yum repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 2.2 components. This command uninstalls the HDP 2.2 components. It leaves the user data, and metadata, but removes your configurations. Refer to the HDP Component Files for SLES11 SP1 table for the names of the files that need to be removed for each component and use the following format:

   ```
   zypper rm "accumulo_$VERSION_$BUILD*" "datafu_$VERSION_$BUILD*" "falcon_
   $VERSION_$BUILD*" "flume_$VERSION_$BUILD*" "hadoop_$VERSION_$BUILD*"
    "hadooplzo_$VERSION_$BUILD*" ...
   ```

   The following table lists the files that need to be deleted for each of the HDP components. Variables are used to indicate:

- $VERSION - HDP version number in the following format: 2_3_0_0
- $HUE-VERSION - Hue version number in the following format : 2.6.1.2.3.0.0
- $BUILD - the HDP build number in the following format : 3518

So, the file name will look like: accumulo-2_3_0_0_3518*

Table 1.5. HDP Component Files for SLES11 SP1

Component	Associated Files
Accumulo	accumulo_$VERSION_$BUILD*
Atlas	atlas_metadata_$VERSION_$BUILD*
Datafu	datafu_$VERSION_$BUILD*
Falcon	falcon_$VERSION_$BUILD*
Flume	flume_$VERSION_$BUILD*
Hadoop (includes hadoop_client and hadoop_hdfs)	hadoop_$VERSION_$BUILD* hadoop_$VERSION_$BUILD-client* hadoop_$VERSION_$BUILD-hdfs* hadooplzo_$VERSION_$BUILD*
HBase	hbase_$VERSION_$BUILD*
Hive	hive_$VERSION_$BUILD* hive_hbase_$VERSION_$BUILD* hive_hcatalog_$VERSION_$BUILD* hive_jdbc_$VERSION_$BUILD* hive_metastore_$VERSION_$BUILD* hive_server_$VERSION_$BUILD* hive_webhcathive_hbase_$VERSION_$BUILD* ranger_hive_$VERSION_$BUILD*
Hue	hue_$HUE_VERSION_$BUILD* hue_beeswax_$HUE_VERSION_$BUILD* hue_common_$HUE_VERSION_$BUILD* hue_hcatalog_$HUE_VERSION_$BUILD* hue_oozie_$HUE_VERSION_$BUILD* hue_pig_$HUE_VERSION_$BUILD* hue_server_$HUE_VERSION_$BUILD*
Kafka	kafka_$VERSION_$BUILD*
Knox	knox_$VERSION_$BUILD*
Mahout	mahout_$VERSION_$BUILD*
MapReduce	hadoop_$VERSION_$BUILD-mapreduce*
Oozie	oozie_$VERSION_$BUILD*
Phoenix	phoenix_$VERSION_$BUILD
Pig	ranger_$VERSION_$BUILD*

Component	Associated Files
Ranger	ranger_$VERSION_$BUILD*
Slider	slider_$VERSION_$BUILD*
Spark	spark_$VERSION_$BUILD*
Sqoop	sqoop_$VERSION_$BUILD*
Storm	storm_$VERSION_$BUILD*
Tez	tez_$VERSION_$BUILD*
YARN	hadoop_$VERSION_$BUILD-yarn*
ZooKeeper	zookeeper_$VERSION_$BUILD*

3. Validate that all HDP 2.2 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.2
   ```

4. Remove your old hdp.repo file:

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

5. Download the HDP 2.3.0 hdp.repo file:

   ```
   wget -nv http://public-repo-1.hortonworks.com/HDP/sles11sp1/2.x/updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.2 components:

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

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

   ```
   zypper up -r HDP-2.3
   ```

   ```
   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

1. On all hosts, clean the zypper repository.

```
zypper clean -all
```

2. Remove your old HDP 2.2 components. This command uninstalls the HDP 2.2 components. It leaves the user data, and metadata, but removes your configurations. Refer to the HDP Component Files for SLES11 SP3/SP4 table for the names of the files that need to be removed for each component and use the following format:

```
zypper rm "accumulo-$VERSION-$BUILD*" "datafu-$VERSION-$BUILD*" "falcon-
$VERSION-$BUILD*" "flume-$VERSION-$BUILD*" "hadoop-$VERSION-$BUILD*"
 "hadooplzo-$VERSION-$BUILD*" ...
```

The following table lists the files that need to be deleted for each of the HDP components. Variables are used to indicate:

- $VERSION - HDP version number in the following format: 2_3_0_0
- $HUE-VERSION - Hue version number in the following format : 2.6.1.2.3.0.0
- $BUILD - the HDP build number in the following format : 3518

So, the file name will look like: accumulo-2_3_0_0_3518*

Table 1.6. HDP Component Files for SLES11 SP3/SP4

Component	Associated Files
Accumulo	accumulo_$VERSION_$BUILD*
Atlas	atlas_metadata_$VERSION_$BUILD*
Datafu	datafu_$VERSION_$BUILD*
Falcon	falcon_$VERSION_$BUILD*
Flume	flume_$VERSION_$BUILD*
Hadoop (includes hadoop_client and hadoop_hdfs)	hadoop_$VERSION_$BUILD* hadoop_$VERSION_$BUILD-client* hadoop_$VERSION_$BUILD-hdfs* hadooplzo_$VERSION_$BUILD*
HBase	hbase_$VERSION_$BUILD*
Hive	hive_$VERSION_$BUILD* hive_hbase_$VERSION_$BUILD* hive_hcatalog_$VERSION_$BUILD* hive_jdbc_$VERSION_$BUILD* hive_metastore_$VERSION_$BUILD* hive_server_$VERSION_$BUILD* hive_webhcathive_hbase_$VERSION_$BUILD* ranger_hive_$VERSION_$BUILD*
Hue	hue_$HUE_VERSION_$BUILD* hue_beeswax_$HUE_VERSION_$BUILD* hue_common_$HUE_VERSION_$BUILD*

Component	Associated Files
	hue_hcatalog_$HUE_VERSION_$BUILD* hue_oozie_$HUE_VERSION_$BUILD* hue_pig_$HUE_VERSION_$BUILD* hue_server_$HUE_VERSION_$BUILD*
Kafka	kafka_$VERSION_$BUILD*
Knox	knox_$VERSION_$BUILD*
Mahout	mahout_$VERSION_$BUILD*
MapReduce	hadoop_$VERSION_$BUILD-mapreduce*
Oozie	oozie_$VERSION_$BUILD*
Phoenix	phoenix_$VERSION_$BUILD
Pig	ranger_$VERSION_$BUILD*
Ranger	ranger_$VERSION_$BUILD*
Slider	slider_$VERSION_$BUILD*
Spark	spark_$VERSION_$BUILD*
Sqoop	sqoop_$VERSION_$BUILD*
Storm	storm_$VERSION_$BUILD*
Tez	tez_$VERSION_$BUILD*
YARN	hadoop_$VERSION_$BUILD-yarn*
ZooKeeper	zookeeper_$VERSION_$BUILD*

3. Validate that all HDP 2.2 component binaries are uninstalled:

   ```
   zypper search --installed-only --repo HDP-2.2.6.0
   ```

4. Remove your old hdp.repo file:

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

5. Download the HDP 2.3.0 hdp.repo file:

   ```
   http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
   updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
   ```

6. Install the HDP 2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.2 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" "ranger" "kafka" "spark"
    "spark-python" "hdp_mon_nagios_addons" "slider" "hive-webcat" "hive-
   hcatalog""
   ```

   ```
   zypper up -r HDP-2.3
   ```

   ```
   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.

1.3. Symlink Directories with hdp-select

Warning

HDP 2.3.0 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 2.3.0.0-<$version>
   ```

   For example:

   ```
   /usr/bin/hdp-select set all 2.3.0.0-2800
   ```

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 2.3.0. 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 will 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 2.3.0. 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 will 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 will 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.

   Run this command on all NameNode hosts:

   ```
   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 will not enter the standby state as usual. Rather, this NameNode will immediately enter 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 will be 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 will download 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 will need to substitute your Service user name in each of the `su` commands.

Important

If you have a secure server, you will 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 will 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/2.3.0.0-<$version>/
   mapreduce/"

   su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hadoop/
   mapreduce.tar.gz /hdp/apps/2.3.0.0-<$version>/mapreduce/"

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

   su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<
   $version>/mapreduce"
   ```

```
su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<
$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>yyarn.app.mapreduce.am.admin-command-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 2.3.0.0-<$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 /usr/hdp/2.3.0.0-<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/bin/container-
executor

chmod 6050 /usr/hdp/${hdp.version}/hadoop-yarn/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 will 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
```

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 will redirect 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 will 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 2.3.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.2 will need to be recalculated to attain identical memory allotments in HDP 2.3.0. The L1 LruBlockCache will be whatever hfile.block.cache.size is set to and the L2 BucketCache will be 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
   ```

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 will 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 will 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 2.3.0.0-2800.

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 `2.3.0.0-2800`.

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.

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 will 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 `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 will 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 from `/usr/share/java` to libext-customer:

   a. Create the `/usr/hdp/2.3.0.0-<version>/oozie-server/libext-customer` directory.

   ```
   cd /usr/hdp/2.3.0.0-<version>/oozie-server

   mkdir libext-customer
   ```

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

   ```
   chmod -R 777 /usr/hdp/2.3.0.0-<version>/oozie-server/libext-customer
   ```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/2.3.0.0-<version>/hadoop-client/lib/
   hadoop*lzo*.jar /usr/hdp/current/oozie-server/libext-customer

   cp /usr/share/HDP-oozie/ext.zip /usr/hdp/2.3.0.0-<version>/
   oozie-server/libext-customer/
   ```

   Also, copy Oozie db jar in libext-customer.

4. If Falcon was also installed and configured before upgrade in HDP 2.2.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"
```

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', '2.5');
```

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. 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.

11. Start Oozie as the Oozie user:

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

12. Check processes.

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

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 Apache WebHCat Service user. If you are using another name for these Service users, you will 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 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/2.3.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/pig/pig.tar.gz /
hdp/apps/2.3.0.0-<$version>/pig/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hive/hive.tar.
gz /hdp/apps/2.3.0.0-<$version>/hive/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/sqoop.tar.
gz /hdp/apps/2.3.0.0-<$version>/sqoop/"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/pig"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/pig/
pig.tar.gz"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/hive"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/hive/
hive.tar.gz"
```

```
su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$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 2.3.0 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

1. Replace your configuration after upgrading. Copy `/etc/pig/conf` from the template to the conf directory in pig hosts.
2. To validate the Apache Pig upgrade, complete the following steps:
   a. On the host machine where Pig is installed, run the following commands:

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

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

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

   c. Run the Pig script:

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

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 will 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/2.3.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.3.0.0-<$version>/
sqoop"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/sqoop.tar.gz /
hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.tar.gz"
```

```
su - hdfs -c "hdfs dfs -chmod 444 /hdp/apps/2.3.0.0-<$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.

4. Because Sqoop is a client tool with no server component, you will need to run your own jobs to validate the upgrade.

1.20. Configure, Start, and Validate Apache Flume

1. If you have not already done so, upgrade Apache Flume. On the Flume host machine, run the following command:

   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum upgrade flume
     ```

   - For **SLES**:

     ```
     zypper update flume
     ```

     ```
     zypper remove flume
     ```

     ```
     zypper se -s flume
     ```

     You should see Flume in the output.

     Install Flume:

     ```
     zypper install flume
     ```

2. To confirm that Flume is working correctly, create an example configuration file. The following snippet is a sample configuration that can be set using the properties file. For more detailed information, see the “Flume User Guide.”

   ```
   agent.sources = pstream
   agent.channels = memoryChannel
   agent.channels.memoryChannel.type = memory

   agent.sources.pstream.channels = memoryChannel
   agent.sources.pstream.type = exec
   agent.sources.pstream.command = tail -f /etc/passwd

   agent.sinks = hdfsSink
   agent.sinks.hdfsSink.type = hdfs
   agent.sinks.hdfsSink.channel = memoryChannel
   agent.sinks.hdfsSink.hdfs.path = hdfs://tmp/flumetest
   agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
   agent.sinks.hdfsSink.hdfs.writeFormat = Text
   ```

   The source here is defined as an exec source. The agent runs a given command on startup, which streams data to stdout, where the source gets it. The channel is defined as an in-memory channel and the sink is an HDFS sink.

3. Given this configuration, you can start Flume as follows:

```
$ bin/flume-ng agent --conf ./conf --conf-file example.conf --name a1 -
Dflume.root.logger=INFO,console
```

Note

The directory specified for `--conf agrument` would include a shell script flume-env.sh and potentially a log4j properties file. In this example, we pass a Java option to force Flume to log to the console and we go without a custom environment script.

4. After validating data in `hdfs://tmp/flumetest`, stop Flume and resore any backup files. Copy `/etc/flume/conf` to the conf directory in Flume hosts.

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 will 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. Enter the mahout command to convert the plain text file sample-test.txt into a sequence file stored in the output directory mahouttest:

```
mahout seqdirectory --input /user/testuser/testdata --output /user/testuser/
mahouttest -ow --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 will need to substitute your Service user name in each of the su commands.

For HDP 2.3.0, use the Hue version shipped with HDP 2.3.0. If you have a previous version of Hue, use the following steps to upgrade Hue.

Complete one of the following:

- **SQLite**

  1. Copy the hue.ini settings from your old hue.ini configuration file to new hue.ini configuration file.

  2. 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
     ```

- **MySQL**

  1. Copy the hue.ini settings from your old hue.ini configuration file to new hue.ini configuration file.

  2. 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 2.2 to 2.3.x, you should also make sure that your individual Hadoop components are also upgraded to the latest version.

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 will need to upgrade Knox manually to the latest version. Because "rolling upgrades" are now supported in HDP 2.3.0, 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.

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

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. Start the ldap service.

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

5. For HDP 2.3.0, the default paths for Knox change. Upgrade Knox in order to update these paths.

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

   b. Start the Gateway:

   ```
   su -l 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 -c "/usr/hdp/{the new HDP server version}/knox-server/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/
   ?op=LISTSTATUS
   ```

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

   ```
   curl -i -v -k -u admin:admin-password -X GET "https://localhost:8443/
   gateway/admin/api/v1/version"
   curl -i -v -k -u admin:admin-password -X GET "https://localhost:8443/
   gateway/sandbox/webhdfs/v1?op=LISTSTATUS"
   ```

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 will need to downgrade the Knox Gateway to the previous version. The steps to downgrade the Knox Gateway are described in the next section.

1.24. Configure and Validate Apache Falcon

Note

In HDP 2.3.0, 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 upgrading 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 2.3.0 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/
   ```

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 will 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 2.3.0 and be on the latest version of Storm.

1. After upgrading Storm, replace your configuration. Copy `/etc/storm/conf` from the template to the `conf` directory .

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

3. 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.

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

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

5. 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

Before you can upgrade the Apache Ranger service, you must have first upgraded your HDP components to the latest version (in this case, 2.3.0). This section assumes that you have already performed the following tasks, however, if you have not already performed these steps, refer to the "Upgrade HDP 2.2 Components" section in this guide for instructions on how to upgrade your HDP components to 2.3.0.

1.26.1. Preparing Your Cluster to Upgrade Ranger

If you are not currently using Ambari to manage your Hadoop cluster, you will need to upgrade 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 xasecure > dest_dir/filename.sql
mysqldump -u root -p root xasecure_audit > dest_dir/filename.sql
```

Note

Review the data in the xa_access_audit table in the Ranger Audit Database. Truncate historical data to reduce upgrading time, especially if you have a lot of data in xa_access_audit table.

1.26.2. Stop the Ranger Services

Now that you have prepared your cluster for the Ranger upgrade, you will 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 will 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 will 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.

1.26.3. Preparing the Cluster for Upgrade

Before you begin the upgrade process, you will 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 will 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

1.26.4. Registering the HDP 2.3 Repo

After you have prepared your cluster for the upgrade, you need to register the HDP 2.3.0 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_3_*-admin
   yum install ranger_2_3_*-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. Update the `install.properties` file to migrate the database credentials properties and POLICYMGR_EXTERNAL-URL property from HDP 2.2. to HDP 2.3.0.

4. 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
```

5. This should successfully install the Ranger Admin component.
6. Start the Ranger Admin component.

```
service ranger-admin start
```

7. You must now configure and setup the Ranger UserSync component by migrating the properties from the HDP 2.2 `install.properties` file (POLICY_MGR_URL, SYNC_SOURCE and LDAP/AD properties.
8. 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
```

9. Start the Ranger UserSync component.

```
service ranger-usersync start
```

1.26.5. Install the Ranger Components

Next, you will 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

Note

When installing each Ranger component, you will also need to make sure you upgrade each individual component to version 2.3.0 before restarting each service.

With this release, Ranger has also added support for the following components:

- Solr
- Kafka
- YARN

1.26.6. Restart the Ranger Services

Once you have re-installed each Ranger component, you will 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 will 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
```

1.26.7. 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 2.3.0 release, and to ensure smooth operation of Ranger services in your cluster.

Note

When you enable each Ranger plugin, be sure to remove all 2.2 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 will remain 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 will 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

1.27. Configuring and Upgrading Apache Spark

To upgrade Apache Spark, start the service and update configurations.

1. Replace the hdp version in $SPARK_HOME/conf/spark-defaults.conf and $SPARK_HOME/conf/java-opts with the current hadoop version.

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

2. Restart the history server:

   ```
   su - spark -c "usr/hdp/current//start-historyserver/sbin/start-history-server.sh"
   ```

3. If you will be running Spark in yarn-client mode, update the following property in /etc/hadoop/conf/mapred-site.xml by substituting ${hdp.version} with the actual HDP version (2.3.2.0.0- $version>).

   ```
   <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>
   ```

4. Restart Spark on YARN in either yarn-cluster mode or yarn-client mode:

yarn-cluster mode:

```
/usr/hdp/current/spark-client/bin/spark-submit --class path.to.your.Class --
master yarn-cluster [options] <app jar> [app
    options]
```

yarn-client mode:

```
./usr/hdp/current/spark-client/bin/spark-shell --master yarn-client
```

5. To enable Spark to work in LzoCodec, add the following information to `/etc/spark/conf/spark-defaults.conf`, when running on client-mode:

   ```
   spark.driver.extraLibraryPath /usr/hdp/current/hadoop-client/
   lib/naive:/usr/hdp/current/hadoop-client/lib/native/Linux-
   amd64-64
   ```

   ```
   spark.driver.extraClassPath /usr/hdp/current/hadoop-client/lib/
   hadoop-lzo-0.6.0.2.3.2.0.0-2492.jar
   ```

1.28. Upgrade Apache Slider

To upgrade Apache Slider, simply upgrade the Slider client.

1. Upgrade Slider client:

   ```
   hdp-select set slider-client 2.3.0.0-<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.

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 2.2.4.0-2633
      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 2.2.2.0-2041
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. 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 will 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 will be discarded!)

2. As the $HDFS_USER, enter:

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

1.31. Install New HDP 2.3 Services

Install new HDP 2.3.0 Services (see the Non-Ambari Cluster Installation Guide):

- Atlas – a low-level service, similar to YARN, that provides metadata services to the HDP platform.
- SmartSense – a next generation subscription model that features upgrade and configuration recommendations.

2. Upgrade from HDP 2.1 to HDP 2.3 Manually

Important

If you installed and manage HDP-2.1 with Ambari, **you must use the Ambari Upgrade Guide** to perform the the HDP-2.1 to HDP-2.3.0 upgrade.

Note

These instructions cover the upgrade between two minor releases, such as HDP-2.1 to HDP-2.3.0. If you need to upgrade between two maintenance releases such as HDP-2.1.2 to 2.1.7, follow the upgrade instructions in the HDP Release Notes.

Starting with HDP-2.2, HDP supports side-by-side installation of HDP 2.2 and subsequent releases, which lets you perform rolling upgrades on your cluster and improve execution times for in-place upgrade. To support side-by-side installation, the HDP package version naming convention for both RPMs and Debs has changed to include the HDP product version. For example, hadoop-hdfs in HDP-2.2.4 is now hadoop_2.2.4.0.hdfs. HDP-2.2 marked the first release where HDP rpms, debs, and directories contained versions in the names to permit side-by-side installations of later HDP releases. To select from the releases you have installed side-by-side, Hortonworks provides hdp-select, a command that lets you select the active version of HDP from the versions you have installed.

However, because HDP-2.1 did not support side-by-side installation, you will upgrade to HDP-2.3.0 in a way very similar to previous minor-version upgrades. Subsequent upgrades after 2.3.0 will be easier, and if you choose to add Ambari to your cluster, you will be able to use the Rolling Upgrade feature.

Important

You cannot perform a side-by-side rolling upgrade from HDP 2.1 to HDP-2.3.0; only upgrade in place is supported. Side-by-side and rolling upgrade support starts with releases HDP 2.2 and above.

This document provides instructions on how to manually upgrade to HDP-2.3.0 from the HDP 2.1 release. It assumes the existing HDP 2.1 was also installed manually.

The HDP packages for a complete installation of HDP-2.3.0 will take about 2.5 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.

Important

You are strongly encouraged to read completely through this entire document before starting the Manual Upgrade process, in order to understand the interdependencies and the order of the steps.

The following steps are required to upgrade from HDP-2.1 to the latest release of HDP-2.3.0 from HDP-2.1:

1. Get Ready to Upgrade
2. Upgrade HDP 2.1 Components
3. Symlink Directories with hdp-select
4. Configure and Start Apache ZooKeeper
5. Configure and Start Hadoop
6. Start HDFS
7. Configure and Start Apache HBase
8. Configure and Start Apache Phoenix
9. Configure and Start Apache Accumulo
10. Configure and Start Apache Tez
11. Configure and Start Apache Hive and Apache HCatalog
12. Configure and Start Apache Oozie
13. Configure and Start Apache WebHCat (Templeton)
14. Configure and Start Apache Pig
15. Configure and Start Apache Sqoop
16. Configure and Start Apache Flume
17. Configure and Validate Apache Mahout
18. Configure and Start Hue
19. Configure and Start Apache Knox
20. Configure and Start Apache Falcon
21. Finalize the Upgrade
22. Install new HDP-2.3.0 services if desired, such as Ranger or Spark

2.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.1 to HDP-2.3.0 versions and adding the new HDP-2.3.0 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-2.3.0 will take up about 2.5 GB of disk space.

The first step is to ensure you keep a backup copy of your HDP 2.1 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 will 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/zookeeper/conf`
   - `/etc/tez/conf`
   - `/etc/storm/conf`
   - Optional - Back up your userlogs directories, `${mapred.local.dir}/userlogs`.

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

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

3. 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. **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.

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

4. Save the namespace by executing the following commands:

   ```
   su - hdfs

   hdfs dfsadmin -safemode enter

   hdfs dfsadmin -saveNamespace
   ```

5. Backup your NameNode metadata.

   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`

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

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

7. 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 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

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

These instructions are provided for your convenience. Please 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_oozie.sql	mysql $dbname < $inputfilename.sql For example: mysql oozie < /tmp/mydir/backup_oozie.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql For example:	sudo -u $username psql $databasename < $inputfilename.sql For example:

Database Type	Backup	Restore

	`sudo -u`
	
	`postgres pg_dump oozie > /tmp/mydir/backup_oozie.sql`
		`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

9. **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: 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

10. Stop all services (including MapReduce) and client applications deployed on HDFS:

Component	Command
Knox	`cd $GATEWAY_HOME su knox -c "bin/gateway.sh stop"`

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

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

YARN	Run this command on all NodeManagers:
	
	`su - yarn -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec && /usr/lib/hadoop-yarn/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/lib/hadoop/libexec && /usr/lib/hadoop-mapreduce/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/lib/hadoop/libexec && /usr/lib/hadoop-yarn/sbin/yarn-daemon.sh --config /etc/hadoop/conf stop resourcemanager"`
	
	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"`

HDFS	On all DataNodes:
	
	`su - hdfs -c "/usr/lib/hadoop/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/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop secondarynamenode”`
	
	On the NameNode host machine(s):
	
	`su - hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop namenode"`

Component	Command

	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/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop zkfc"`
	
	If you are running NameNode HA, stop the JournalNodes by executing these commands on the JournalNode host machines:
	
	`su - hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/conf stop journalnode"`

ZooKeeper Host machines	`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"`

Ranger (XA Secure)	`service xapolicymgr stop`
	
	`service uxugsync stop`

11. 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 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`

12. 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 will print 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 will then suffix 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` will rename all applicable existing files in the cluster. This may impact cluster applications.

13. 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 2.1 Components

The upgrade process to HDP-2.3.0 involves the following steps. See the HDP-2.3.0 Release Notes for repo information.

RHEL/CentOS/Oracle 6

1. On all hosts, clean the yum repository.

   ```
   yum clean all
   ```

2. Remove your old HDP 2.1 components. This command uninstalls the HDP 2.1 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*"
    "hdp_mon_nagios_addons"
   ```

3. Validate that all HDP 2.1 component binaries are uninstalled:

```
yum list installed | grep @HDP2.1
```

4. Remove your old hdp.repo file:

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

5. Install the HDP-2.3.0 repo:

   - Download the hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/updates/2.3.
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-2.3.0 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-2.3.0 Hortonworks Data Platform Version - HDP-2.3.0
```

6. Install the HDP-2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 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"
 "hdp_mon_nagios_addons"
```

Note

If you installed Apache Argus, it is now 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 2.1 components. This command uninstalls the HDP 2.1 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*"
 "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.1 component binaries are uninstalled:

   ```
   yum list installed | grep @HDP2.1
   ```

4. Remove your old hdp.repo file:

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

5. Install the HDP-2.3.0 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/updates/2.3.
     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-2.3.0 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-2.3.0 Hortonworks Data Platform Version - HDP-2.3.0
     ```

6. Install the HDP-2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 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"
    "hdp_mon_nagios_addons"
   ```

   

   Note

   If you installed Apache Argus, it is now 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 SP 1

1. On all hosts, clean the yum repository.

```
zypper clean -all
```

2. Remove your old HDP 2.1 components. This command uninstalls the HDP 2.1 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*"
 "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.1 component binaries are uninstalled:

```
yum list installed | grep @HDP2.1
```

4. Remove your old hdp.repo file:

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

5. Download the HDP-2.3.0 hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/sles11sp1/2.x/updates/2.3.
0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
```

6. Install the HDP-2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 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"
 "hdp_mon_nagios_addons"
```

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

```
zypper up -r HDP-2.3
```

```
zypper install oozie-client
```

Note

If you installed Apache Argus, it is now 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 hcatlog:

```
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

1. On all hosts, clean the yum repository.

```
zypper clean -all
```

2. Remove your old HDP 2.1 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*" "hdp_mon_nagios_addons"
```

3. Validate that all HDP 2.1 component binaries are uninstalled:

```
yum list installed | grep @HDP2.1
```

4. Remove your old hdp.repo file:

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

5. Download the HDP-2.3.0 hdp.repo file:

```
http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
```

6. Install the HDP-2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 components:

```
zypper 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"
 "hdp_mon_nagios_addons"
```

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

```
zypper up -r HDP-2.3
```

```
zypper install oozie-clientt
```

Note

If you installed Apache Argus, it is now 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.

2.3. Symlink Directories with hdp-select

Warning

HDP-2.3.0 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 2.3-<$version>
   ```

   For example:

   ```
   /usr/bin/hdp-select set all 2.3-2800
   ```

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

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-2.3.0. 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 will 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 in HDP-2.3.0. 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 will 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 will 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 will not enter the standby state as usual. Rather, this NameNode will immediately enter 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 will be 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 will download 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 will need to substitute your Service user name in each of the `su` commands.

Important

If you have a secure server, you will 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 will need to substitute your Service user name in each of the su commands.

Important

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

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

```
su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/
mapreduce/"

su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hadoop/
mapreduce.tar.gz /hdp/apps/2.3.0.0-<$version>/mapreduce/"

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

su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<
$version>/mapreduce"

su - hdfs -c "hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<
$version>/mapreduce/mapreduce.tar.gz"
```

2. Make the following changes to /etc/hadoop/conf/mapred-site.xml:

   - Add:

```
<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}.

- Remove the following properties from `/etc/hadoop/conf/mapred-site.xml`: `mapreduce.task.tmp.dir`, `mapreduce.job.speculative.slownodethreshold` (deprecated), and `mapreduce.job.speculative.speculativecap` (deprecated).

3. Add the following properties to `/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. Modify the following property to `/etc/hadoop/conf/yarn-site.xml`:

```
<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/*,
   $PWD/mr-framework/hadoop/share/hadoop/share/hadoop/tools/lib/*,
  /usr/hdp/${hdp.version}/hadoop/lib/hadoop-lzo-0.6.0.${hdp.version}.jar,
  /etc/hadoop/conf/secure</value>
</property>
```

6. Make the following change to the `/etc/hadoop/conf/yarn-env.sh`:

   Change `export HADOOP_YARN_HOME=/usr/lib/hadoop-yarn`

   to

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

7. Make the following change to the `/etc/hadoop/conf/yarn-env.sh`:

   Change

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

   to

   ```
   HADOOP_LIBEXEC_DIR=/usr/hdp/current/hadoop-client/libexec/
   ```

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

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

     ```
     yarn.nodemanager.linux-container-executor.group=hadoop
     ```

```
banned.users=hdfs,yarn,mapred
min.user.id=1000
```

- 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 will need to substitute your Service user name for "yarn" or "mapreduce" in each of the `su` commands.

1. If you have a highly available HDFS cluster configuration, 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 will redirect 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 will 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-2.3.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.1 will need to be recalculated to attain identical memory allotments in HDP-2.3.0. The L1 LruBlockCache will be whatever hfile.block.cache.size is set to and the L2 BucketCache will be 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 psql.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 will need to substitute your Accumulo Service user name for "accumulo" in each of the `su` commands.

Upon upgrade from HDP 2.1 to HDP-2.3.0, Accumulo automatically changes the HDFS and ZooKeeper data stored on Accumulo. This change is not backward compatible, and HDP 2.1 will not run on this updated data.

After upgrading from HDP 2.1 to HDP-2.3.0, Accumulo automatically upgrades the internal metadata, notably the data in ZooKeeper, when the Accumulo Master for HDP-2.3.0 first starts. This change is not backward compatible and HDP 2.1 will no longer run against the data.

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

2. In HDP-2.3.0, the instance.dfs.dir and instance.dfs.uri properties are deprecated with the instance.volumes property. If it does not already exist, add the instance.volumes property to the accumulo-site.xml file. Do **not** remove the instance.dfs.dir and instance.dfs.uri properties. You can extrapolate the value for the instance.volumes property from the instance.dfs.dir and instance.dfs.uri properties.

   For example:

   ```
   <property>
      <name>instance.dfs.dir</name>
      <value>/accumulo</value>
   </property>
   <property>
      <name>instance.dfs.uri</name>
      <value>hdfs://my_namenode:8020</value>
   ```

```
</property>
<property>
   <name>instance.volumes</name>
   <value>hdfs://my_namenode:8020/accumulo</value>
</property>
```

3. 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"
```

4. 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 will need to substitute your Service user name in each of the `su` commands.

Important

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

To upgrade Apache Tez:

1. Copy your previously backed-up copy of tez-site.xml into the `/etc/tez/conf` directory.

2. Check to see if the HDP-2.3.0 Tez tarball libraries are already in the `/hdp/apps/<hdp 2.3>/tez/` version> directory - if so, then skip this step. If not, put the HDP-2.3.0 Tez tarball libraries in the `/hdp/apps` directory in HDFS, so that submitted Tez applications to this cluster will have access to these shared Tez tarball libraries. Execute this step on any host that has the Tez client installed.

   ```
   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 2.3.0-2800.

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 2.3.0-2800.

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. Update the following tez-site.xml properties to their new names.

Old Property Name	New Property Name
tez.am.java.opts	tez.am.launch.cmd-opts
tez.am.env	tez.am.launch.env
tez.am.shuffle-vertex-manager.min-src-fraction	tez.shuffle-vertex-manager.min-src-fraction
tez.am.shuffle-vertex-manager.max-src-fraction	tez.shuffle-vertex-manager.max-src-fraction
tez.am.shuffle-vertex-manager.enable.auto-parallel	tez.shuffle-vertex-manager.enable.auto-parallel
tez.am.shuffle-vertex-manager.desired-task-input-size	tez.shuffle-vertex-manager.desired-task-input-size
tez.am.shuffle-vertex-manager.min-task-parallelism	tez.shuffle-vertex-manager.min-task-parallelism
tez.am.grouping.split-count	tez.grouping.split-count
tez.am.grouping.by-length	tez.grouping.by-length
tez.am.grouping.by-count	tez.grouping.by-count
tez.am.grouping.max-size	tez.grouping.max-size
tez.am.grouping.min-size	tez.grouping.min-size
tez.am.grouping.rack-split-reduction	tez.grouping.rack-split-reduction
tez.am.am.complete.cancel.delegation.tokens	tez.cancel.delegation.tokens.on.completion
tez.am.max.task.attempts	tez.am.task.max.failed.attempts
tez.generate.dag.viz	tez.generate.debug.artifacts
tez.runtime.intermediate-output.key.comparator.class	tez.runtime.key.comparator.class
tez.runtime.intermediate-output.key.class	tez.runtime.key.class
tez.runtime.intermediate-output.value.class	tez.runtime.value.class
tez.runtime.intermediate-output.should-compress	tez.runtime.compress
tez.runtime.intermediate-output.compress.codec	tez.runtime.compress.codec
tez.runtime.intermediate-input.key.secondary.comparator.class	tez.runtime.key.secondary.comparator.class
tez.runtime.broadcast.data-via-events.enabled	tez.runtime.transfer.data-via-events.enabled

Old Property Name	New Property Name
tez.runtime.broadcast.data-via-events.max-size	tez.runtime.transfer.data-via-events.max-size
tez.runtime.shuffle.input.buffer.percent	tez.runtime.shuffle.fetch.buffer.percent
tez.runtime.task.input.buffer.percent	tez.runtime.task.input.post-merge.buffer.percent
tez.runtime.job.counters.max	tez.am.counters.max.keys
tez.runtime.job.counters.group.name.max	tez.am.counters.group-name.max.keys
tez.runtime.job.counters.counter.name.max	tez.am.counters.name.max.keys
tez.runtime.job.counters.groups.max	tez.am.counters.groups.max.keys
tez.task.merge.progress.records	tez.runtime.merge.progress.records
tez.runtime.metrics.session.id	tez.runtime.framework.metrics.session.id
tez.task.scale.memory.additional.reservation.fraction.per-io	tez.task.scale.memory.additional-reservation.fraction.per-io
tez.task.scale.memory.additional.reservation.fraction.max	tez.task.scale.memory.additional-reservation.fraction.max
tez.task.initial.memory.scale.ratios	tez.task.scale.memory.ratios
tez.resource.calculator.process-tree.class	tez.task.resource.calculator.process-tree.class

For more information on setting Tez configuration parameters in HDP-2.3.0, see Installing and Configuring Tez in the Non-Ambari Cluster Installation Guide.

6. Change the value of the tez.tez-ui.history-url.base property to the url for the upgraded Tez View. For more details, see Deploying the Tez View in the HDP Ambari Views Guide.

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 will 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.

Important

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

Note

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

```
sudo <POSTGRES_USER>
ALTER DATABASE <HIVE-METASTORE-DB-NAME> OWNER TO <HIVE_USER>
```

Note

If you are using Oracle 11, you might 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.

Note

Copy only the necessary configuration files. Do not copy the env.sh files, for example, hadoop-env.sh, hive-env.sh, and so forth. Additionally, all env.sh files must be properly configured.

4. Edit the hive-site.xml file and modify the properties based on your environment.

   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 `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 will 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-server/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/lib/hadooplzo*.jar/usr/hdp/current/
oozie-server/libext-customer

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

4. 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"
```

5. If a previous version of Oozie was created using auto schema creation, you must run an SQL query.

Use the `oozie-site.xml` properties:

- `oozie.service.JPAService.jdbc.username`

- `oozie.service.JPAService.jdbc.username`

- `oozie.service.JPAService.jdbc.url`

to obtain the password, username and db to run the query.

Run the SQL query:

```
insert into oozie_sys (name, data) values ('db.version', '2.5');
```

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

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

7. 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/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
```

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

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

9. Add the following property to 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. Upgrade the Oozie database:

```
su - oozie -c "bin/ooziedb.sh upgrade -run"
```

11. Start Oozie as the Oozie user:

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

12. Check processes.

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

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 Apache WebHCast Service user. If you are using another name for these Service users, you will 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 WebHCat configuration files.

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

```
su - hdfs -c "dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/pig/"

su - hdfs -c "dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/hive/"

su - hdfs -c "dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop/"

su - hdfs -c "dfs -put /usr/hdp/2.3.0.0-<$version>/pig/pig.tar.gz /hdp/
apps/2.3.0.0-<$version>/pig/"

su - hdfs -c "dfs -put /usr/hdp/2.3.0.0-<$version>/hive/hive.tar.gz /hdp/
apps/2.3.0.0-<$version>/hive/"

su - hdfs -c "dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/sqoop.tar.gz /
hdp/apps/2.3.0.0-<$version>/sqoop/"

su - hdfs -c "dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/pig"

su - hdfs - "dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/pig/pig.tar.
gz"

su - hdfs -c "dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/hive"

su - hdfs -c "dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/hive/hive.
tar.gz"

su - hdfs -c "dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.
tar.gz"

su - hdfs -c "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. 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>
```

d. 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-2.3.0 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. At the WebHCat host machine, run the following command:

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

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

```
curl --negotiate -u: http://cluster.$PRINCIPAL.$REALM:50111/
templeton/v1/status {"status":"ok","version":"v1"}
[machine@acme]$
```

2.18. Configure Apache Pig

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

2. To validate the Apache Pig upgrade, complete the following steps:

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

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

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

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

   c. Run the Pig script:

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

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 will 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 "dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "dfs -chown -R hdfs:hadoop /hdp/apps/2.3.0.0-<$version>/sqoop"

su - hdfs -c "dfs -put /usr/hdp/2.3-<$version>/sqoop/sqoop.tar.gz /hdp/apps/
2.3.0.0-<$version>/sqoop/sqoop.tar.gz"

su - hdfs -c "dfs -chmod 444 /hdp/apps/2.3.0.0-<$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.
4. Because Sqoop is a client tool with no server component, you will need to run your own jobs to validate the upgrade.

2.20. Configure, Start, and Validate Apache Flume

1. If you have not already done so, upgrade Apache Flume. On the Flume host machine, run the following command:

   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum upgrade flume
     ```

   - For **SLES**:

     ```
     zypper update flume

     zypper remove flume

     zypper se -s flume
     ```

     You should see Flume in the output.

     Install Flume:

```
zypper install flume
```

2. To confirm that Flume is working correctly, create an example configuration file. The following snippet is a sample configuration that can be set using the properties file. For more detailed information, see the "Flume User Guide."

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory

agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd

agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://tmp/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

   The source here is defined as an exec source. The agent runs a given command on startup, which streams data to stdout, where the source gets it. The channel is defined as an in-memory channel and the sink is an HDFS sink.

3. Given this configuration, you can start Flume as follows:

```
$ bin/flume-ng agent --conf ./conf --conf-file example.conf --name a1 -
Dflume.root.logger=INFO,console
```

Note

The directory specified for `--conf agrument` would include a shell script flume-env.sh and potentially a log4j properties file. In this example, we pass a Java option to force Flume to log to the console and we go without a custom environment script.

4. After validating data in `hdfs://tmp/flumetest`, stop Flume and resore any backup files. Copy `/etc/flume/conf` to the conf directory in Flume hosts.

2.21. Configure 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 will need to substitute your Service user name in each of the `su` commands.

Replace your configuration after upgrading. Copy `/etc/mahout/conf` from the backup if it existed 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. Enter the mahout command to convert the plain text file sample-test.txt into a sequence file stored in the output directory mahouttest:

```
mahout seqdirectory --input /user/testuser/testdata --output /user/testuser/
mahouttest -ow --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 will need to substitute your Service user name in each of the `su` commands.

For HDP-2.3.0, use the Hue version shipped with HDP-2.3.0. 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 - hdfs
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
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 2.1 to 2.2.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. 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 will be 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 will need to upgrade Knox manually to the latest version. Because “rolling upgrades” are now supported in HDP-2.3.0, 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.
2. Stop each Knox server.

```
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.

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

4. 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.
5. Create the Master Secret:

```
su - knox -c "/usr/hdp/current/knox/bin/knoxcli.sh create-
master"
```

6. Start the Gateway:

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

7. 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 will 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 will 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-2.3.0, if authorization is enabled (for example, in the startup.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-2.3.0 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/
   ```

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 will 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-2.3.0 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-2.3.0 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
```

7. Replace your configuration after upgrading. Copy /etc/storm/conf from the template to the conf directory .
8. Replace your ZooKeeper configuration after upgrading. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.
9. Start ZooKeeper. 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 nimbus, then supervisor/ui/drpc/logviewer:

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

11. 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

Before you can upgrade the Apache Ranger service, you must have first upgraded your HDP components to the latest version (in this case, 2.3.0). This section assumes that you

already have already performed the following tasks, however, if you have not already performed these steps, refer to the "*Upgrade HDP 2.1 Components*" section in this guide for instructions on how to upgrade your HDP components to 2.3.0.

Note

XA Secure was an add-on component in HDP-2.1. Ranger is the new name for XA Secure. In HDP-2.2 and subsequent releases, Ranger is installed with HDP.

2.26.1. Preparing Your Cluster to Upgrade Ranger

If you are not currently using Ambari to manage your Hadoop cluster, you will need to upgrade 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

     ```
     /usr/lib/xapolicymgr
     ```

   - Ranger UserSync

     ```
     /usr/lib/uxugsync or /etc/uxugsync (Depending on your installation)
     ```

   - 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

2.26.2. Stop the Ranger Services

Now that you have prepared your cluster for the Ranger upgrade, you will 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 will receive an acknowledgement from the server that the service has been stopped.

   ```
   service xapolicymgr stop
   ```

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

   ```
   service uxugsync stop
   ```

3. Stop each individual Ranger plugin (HDFS, HBase, Knox, Storm). You will receive an acknowledgement from the server that the plugin has been stopped.

   ```
   service <plugin name> stop
   ```

2.26.3. Install the Ranger Components

Next, you will 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. The following components must be installed:

- Ranger Policy Admin service
- Ranger UserSync service
- Ranger Plugins:
  - HDFS
  - HBase
  - Hive
  - Knox
  - Storm

With this release, Ranger has also added support for the following components:

- Solr
- Kafka
- YARN

2.26.4. Restart the Ranger Services

Once you have re-installed each Ranger component, you will then need to restart these components to ensure the new configurations are loaded in your cluster. 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
```

2.26.5. Remove Existing Startup Files and Symbolic Links

In order to ensure that your Ranger components are upgraded correctly, and there are no conflicts between versions, you should remove any existing startup files and symbolic links from the previous Ranger 2.2 version. The steps you need to follow to remove these files and links are described below.

1. Remove the Policy Manager startup files.

```
rm -f /etc/init.d/xapolicymgr
```

2. Remove the Policy Manager symbolic links.

```
rm -rf /etc/rc*.d/*xapolicymgr
```

3. Remove the UserSync startup files.

```
rm -f /etc/rc*.d/*uxugsync
```

4. Remove the UserSync symbolic links.

```
rm -rf /etc/rc*.d/uxugsync
```

5. Remove the Policy Manager library files.

```
rm -f /usr/lib/xapolicymgr
```

6. Remove the UserSync library files.

```
rm -f /usr/lib/uxugsync
```

2.26.6. 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 2.3.0 release.

Note

When you enable each Ranger plugin, make sure you remove all 2.1 class name values.

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 will 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

2.27. 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 will 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 filesystem health before finalizing the upgrade. (After you finalize an upgrade, your backup will be discarded!)
2. As the $HDFS_USER, enter:

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

2.28. Install New HDP 2.3 Services

Install new HDP-2.3.0 Services (see the Non-Ambari Cluster Installation Guide):

- Atlas – a low-level service, similar to YARN, that provides metadata services to the HDP platform.
- SmartSense – a next generation subscription model that features upgrade and configuration recommendations.

3. Upgrade from HDP 2.0 to HDP 2.3 Manually

HDP 2.3 supports side-by-side installation of HDP 2.2 and above releases, which lets you perform rolling upgrades on your cluster and improve execution times for in-place upgrade. To support side-by-side installation, the HDP package version naming convention for both RPMs and Debs has changed to include the HDP product version. For example, hadoop-hdfs is now hadoop-2.3.6.0-hdfs. HDP 2.2 marks the first release where HDP rpms, debs, and directories contain versions in the names to permit side-by-side installations of later HDP releases.

To select from the releases you have installed side-by-side, Hortonworks provides hdp-select, a command that lets you select the active version of HDP from the versions you have selected.

HDP packages for a complete installation of HDP 2.3 will take about 2.5 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.

This document provides instructions on how to upgrade to HDP 2.3 from the HDP 2.0 release.

Note

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.

1. Download HDP 2.3
2. Get Ready to Upgrade
3. Configure and Start Hadoop
4. Start HDFS
5. Upgrade Apache ZooKeeper
6. Upgrade Apache HBase
7. Upgrade Apache Hive and Apache HCatalog
8. Upgrade Apache Oozie
9. Upgrade Apache WebHCat (Templeton)
10. Upgrade Apache Pig

11. Upgrade Apache Sqoop
12. Upgrade Apache Flume
13. Upgrade Apache Mahout
14. Upgrade Hue
15. Finalize the Upgrade
16. Install additional HDP 2.3 services

3.1. Getting Ready to Upgrade

HDP Stack upgrade involves upgrading from HDP 2.0 to HDP 2.3 versions and adding the new HDP 2.3 services.

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 will need to substitute your Service user name in each of the `su` commands.

The first step is to ensure you keep a backup copy of your HDP 2.0 configurations.

Note

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

1. 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 2.3 will take up about 2.5 GB of disk space.

2. Back up the following HDP directories:

   - /etc/hadoop/conf
   - /etc/hbase/conf
   - /etc/hive/conf
   - /etc/pig/conf
   - /etc/sqoop/conf
   - /etc/flume/conf
   - /etc/mahout/conf
   - /etc/oozie/conf
   - /etc/zookeeper/conf

- **Optional:** Back up your userlogs directories, ${mapred.local.dir}/userlogs.

3. 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"
   ```

4. 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 second command does a recursive listing of the root file system.)

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

   c. **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.

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

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

   ```
   su - hdfs

   hdfs dfsadmin -safemode enter

   hdfs dfsadmin -saveNamespace
   ```

6. Back up your NameNode metadata.

   - Copy the following checkpoint files into a backup directory. If these directories do not exist, create them.

     - `dfs.namenode.dir/edits`

       `dfs.namenode.name.dir/image/fsimage`

       `dfs.namenode.name.dir/current/fsimage`

   - Store the layoutVersion of the namenode.

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

7. Finalize any **PRIOR** HDFS upgrade, if you have not done so already.

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

8. **Optional:** Back up the Hive Metastore database.

   The following instructions are provided for your convenience. For the latest backup instructions, please 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	Connect to the Oracle database using sqlplus export the database: exp username/password@database full=yes file=output_file.dmp mysql $dbname < $inputfilename.sqlsbr For example: mysql hive < /tmp/mydir/backup_hive.sql	Import the database: imp username/password@database file=input_file.dmp

9. Optional: Back up the Oozie metastore database.

   These instructions are provided for your convenience. Please check your database documentation for the latest back up 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_oozie.sql	mysql $dbname < $inputfilename.sql For example: mysql oozie < /tmp/mydir/backup_oozie.sql
Postgres	sudo -u $username pg_dump $databasename > $outputfilename.sql For example: sudo -u postgres pg_dump oozie > /tmp/mydir/backup_oozie.sql	sudo -u $username psql $databasename < $inputfilename.sql For example: sudo -u postgres psql oozie < /tmp/mydir/backup_oozie.sql

10. **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

11. Stop all services (including MapReduce) and client applications deployed on HDFS using the instructions provided in the Stopping HDP Services.

12. 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 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
```

13. 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 will print 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 will suffix 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` will rename all applicable existing files in the cluster. This may impact cluster applications.

3.2. Upgrade HDP 2.0 Components

The upgrade process to HDP 2.3 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 2.0 components. This command uninstalls the HDP 2.0 components. It leaves the user data, and metadata, but removes your configurations:

   ```
   yum erase "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
   "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*"
   "flume*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell"
   "hdp_mon_nagios_addons"
   ```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

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

   - Confirm that the HDP repository is configured.

     ```
     yum repolist
     ```

     You should see something like this. Verify that you have the HDP-2.3.0.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.3.0.0 Hortonworks Data Platform Version - HDP-2.3.0.0
     ```

5. Install the HDP 2.3.0 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.2 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"
   ```

6. Verify that the components were upgraded.

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

   Check to make sure no component file names from the previous stack 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 2.0 components. This command uninstalls the HDP 2.0 components. It leaves the user data, and metadata, but removes your configurations:

```
yum erase "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
"gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*" "hive*"
"flume*" "accumulo*" "mahout*" "hue" "hue-common" "hue-shell"
"hdp_mon_nagios_addons"
```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/
     updates/2.3.0.0/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-2.3.0.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.3.0.0 Hortonworks Data Platform Version - HDP-2.3.0.0
     ```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 components:

```
yum install "hadoop" "hadooplzo" "webhcat" "hcatalog" "oozie"
"collectd" "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive"
"flume" "accumulo" "mahout" "hue" "hdp_mon_nagios_addons"
```

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

6. 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 SP 1

1. On all hosts, clean the yum repository.

```
zypper clean all
```

2. Remove your old HDP 2.1 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*"
"hdp_mon_nagios_addons"
```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/
sles11sp1/2.x/updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/
hdp.repo
```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.0 components:

```
zypper install "hadoop" "hadooplzo" "webhcat" "oozie" "collectd"
"gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
"falcon" "flume" "phoenix" "accumulo" "mahout" "hue" "knox"
"hdp_mon_nagios_addons"

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

zypper up -r HDP-2.3.0.0

zypper install oozie-client
```

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

```
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

1. On all hosts, clean the yum repository.

```
zypper clean all
```

2. Remove your old HDP 2.1 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*"
"hdp_mon_nagios_addons"
```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

     ```
     http://public-repo-1.hortonworks.com/HDP/suse11sp3/2.x/
     updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/hdp.repo
     ```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 2.1 components:

   ```
   zypper install "hadoop" "hadooplzo" "webhcat" "oozie" "collectd"
   "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "tez" "storm"
   "falcon" "flume" "phoenix" "accumulo" "mahout" "hue" "knox"
   "hdp_mon_nagios_addons"

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

   zypper up -r HDP-2.3.0.0

   zypper install oozie-client
   ```

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

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

   No component files names should appear in the returned list.

3.3. Symlink Directories with hdp-select

Warning

HDP 2.3 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.

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

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

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

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

  For example:

```
/usr/bin/hdp-select set all 2.3.0.0-2
```

3.4. Configure and Start Apache ZooKeeper

Tip

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

Note

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

1. Replace your configuration after upgrading. Replace the ZooKeeper template configuration in /etc/zookeeper/conf.
2. Start ZooKeeper.

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

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

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

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 will 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 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 will not load, as this is not where lzo is installed.

3.6. Set RPC Authentication

To successfully upgrade to HDP 2.3, you must change the value of hadoop.rpc.protection to authentication in core-site.xml:

```
<property>
 <name>hadoop.rpc.protection</name>
 <value>authentication</value>
</property>
```

3.7. Set RPC Authentication

To successfully upgrade to HDP 2.3, you must change the value of hadoop.rpc.protection to authentication in core-site.xml:

```
<property>
 <name>hadoop.rpc.protection</name>
 <value>authentication</value>
</property>
```

3.8. Start Hadoop Core

Warning

Before you start HDFS on an HA system 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 will need to substitute your Service user name in each of the `su` commands.

1. If you are upgrading from an HA NameNode configuration, start all JournalNodes. On each JournalNode host, run the following command:

   ```
   su - hdfs -c "/usr/hdp/current/hadoop-hdfs-journalnode/../hadoop/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.

2. 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 command:

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

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

Note

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

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

   In a NameNode HA configuration, this NameNode will not enter the standby state as usual. Rather, this NameNode will immediately enter 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 will be 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 will download 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.

3. Verify that the NameNode is up and running:

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

4. Start the Secondary NameNode. On the Secondary NameNode host machine, run the following command:

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

5. Verify that the Secondary NameNode is up and running:

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

6. Start DataNodes.

   On each of the DataNodes, enter the following command. 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-hdfs-datanode/../hadoop/
   sbin/hadoop-daemon.sh --config /etc/hadoop/conf start datanode"
   ```

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

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

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

   ```
   hdfs dfsadmin -safemode wait

   Safemode is OFF
   ```

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

3.9. Verify HDFS Filesystem Health

Analyze if the filesystem is healthy.

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

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

2. Run hdfs namespace and report.

   a. List directories.

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

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

      ```
      hdfs dfsadmin -report > dfs-new-report-1.log
      ```

3. Compare the namespace report from before the upgrade with the report after the upgrade. Verify that user files exist after upgrade:

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

   

   Note

   You must do this comparison manually to catch all errors.

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

```
http://<namenode>:50070
```

3.10. Configure YARN and MapReduce

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

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

```
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/mapreduce/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hadoop/mapreduce.tar.gz /hdp/apps/
2.3.0.0-<$version>/mapreduce/
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/mapreduce
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/mapreduce/mapreduce.tar.
gz
```

2. Make the following changes to mapred-site.xml:

   - Add:

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

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

<property>
 <name>mapreduce.application.classpath</name>
```

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

3. Add the following property to 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. For secure clusters, you must create and configure the container-executor.cfg configuration file:

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

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

     - 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
```

3.11. Start YARN/MapReduce Services

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 will need to substitute your Service user name in each of the su commands.

Application Timeline Server has changed. If you installed ATS with HDP 2.0, you need to add the following properties to yarn-site.xml to continue using ATS:

```
yarn.timeline-service.leveldb-timeline-store.path=/var/log/hadoop-yarn/
timeline
yarn.timeline-service.leveldb-timeline-store.ttl-interval-ms=300000

** If you are upgrading to HDP 2.1.3 or higher, use the following setting: **
yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.timeline.
LeveldbTimelineStore

** If you are upgrading to HDP 2.1.2, use the following setting: **
yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.
applicationhistoryservice.timeline.LeveldbTimelineStore

yarn.timeline-service.ttl-enable=true
yarn.timeline-service.ttl-ms=2678400000
yarn.timeline-service.generic-application-history.store-class=org.apache.
hadoop.yarn.server.applicationhistoryservice.NullApplicationHistoryStore
yarn.timeline-service.webapp.address={PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:8188
yarn.timeline-service.webapp.https.address=
{PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:8190
yarn.timeline-service.address={PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:10200
```

Add the following properties to hive-site.xml:

```
hive.execution.engine=mr
hive.exec.failure.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.pre.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.tez.container.size={map-container-size}

** If mapreduce.map.memory.mb > 2GB then set it equal to mapreduce.map.memory.

Otherwise, set it equal to mapreduce.reduce.memory.mb **

hive.tez.java.opts=-server -Xmx800m -Djava.net.preferIPv4Stack=true -
XX:NewRatio=8
-XX:+UseNUMA -XX:+UseParallelGC
```

Use configuration values appropriate for your environment. For example, the value "800" in the preceding example is an example, not a requirement.

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

1. If you are using an HA enabled system, you must upgrade the ZooKeeper service and it must be running.
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 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
   ```

4. To start MapReduce, run the following commands:

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

   start historyserver"

   ps -ef | grep -i jobhistoryserver
   ```

3.12. 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 works 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. In your console log for the MapReduce job, look for a line in this format:

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

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

3.13. Configure and Start Apache HBase

Note

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

1. Start the ZooKeeper Server and NameNode services.
2. As the HBase user, run an upgrade:

```
su - hbase -c "hbase upgrade -execute"
```

   You should see a completed Znode upgrade with no errors.

3. Replace your configuration after upgrading. Replace the HBase template configuration in `/etc/hbase/conf`.
4. Start services. Run as root and suppose $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"
```

5. Check processes.

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

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

3.14. Configure and Start Apache Hive and Apache HCatalog

Warning

In HDP 2.1.3 (Apache Hive 0.13.0) the Decimal data type is now treated as the type Decimal(10,0): 10 digits of precision and 0 scale. This is a change from the variable precision and scale that was available in Hive 0.11.0 and Hive 0.12.0, which allowed up to 38 digits of precision and unlimited scale.

To avoid unintended "rounding" of decimal data, sites that were previously running Hive 0.11.0 and Hive 0.12.0 may need to migrate tables with Decimal columns after upgrading to Hive 0.13.0. For details, see the Apache Hive wiki. For assistance with upgrades that involve Decimal data, please contact Hortonworks Support.

Note

The `su` commands in this section use keywords to represent the Service user. For example, "hive" is used to represent the Hive Service user. If you are using another name for your Service users, you will need to substitute your Service user name 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. Ensure that the required driver .jar file (`oracle.jdbc.driver.OracleDriver`) is available in the `/hive/metastore/lib` folder.

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

   ```
   /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>, run the following commands:

```
su - <POSTGRES_USER>

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. Download and extract the HDP companion files.

   Copy the hive-site.xml file in the configuration_files/hive directory of the extracted companion files to the etc/hive/conf directory on your Hive host machine. This new version of hive-site.xml contains new properties for HDP 2.3.0 features.

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

   - Edit the connection properties for your Hive metastore database in hive- site.xml:

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

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

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

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

   - 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>
```

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

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

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

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

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

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

  (To enable HiveServer2, leave this property value empty.)

```
<property>
 <name>hive.metastore.uris</name>
 <value>thrift://$metastore.server.full.hostname:9083</value>
 <description>URI for client to contact metastore server. sbrTo 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.

- Disable autocreation of schemas to match HDP 2.1+ configurations. Edit `hive-site.xml` to set the value of `datanucleus.autoCreateSchema` to false.

```
<property>
 <name>datanucleus.autoCreateSchema</name>
 <value>false</value>
```

```
 <description>Creates necessary schema on a startup if one doesn't exist.
</description>
</property>
```

6. 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 &"
   ```

7. Start Hive Server2.

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

   ```
   su - hive -c "/usr/hdp/current/hive-server2/bin/hiveserver2 >/
   var/log/hive/hiveserver2.out 2> /var/log/hive/hiveserver2.log &"
   ```

3.15. Configure and Start Apache Oozie

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 will need to substitute your Service user name in each of the `su` commands.

Upgrading Apache Oozie is a complex process. Although the instructions are straight-forward, 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/2.3.0.0-<version>/oozie-server/libext-customer` directory.

      ```
      cd /usr/hdp/2.3.0.0-<version>/oozie-server

      mkdir libext-customer
      ```

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

      ```
      chmod -R 777 /usr/hdp/2.3.0.0-<version>/oozie-server/libext-
      customer
      ```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/2.3.0.0-<version>/hadoop-client/lib/
   hadoop*lzo*.jar /usr/hdp/current/oozie-server/libext-customer
   ```

```
cp /usr/share/HDP-oozie/ext.zip /usr/hdp/2.3.0.0-<version>/
oozie-server/libext-customer/
```

Also, copy Oozie db jar in libext-customer.

4. Extract share-lib.

```
cd /usr/hdp/2.3.0.0-<version>/oozie

tar xzvf /usr/hdp/2.3.0.0-<version>/oozie/oozie-sharelib.tar.gz

su - hdfs -c "hdfs dfs -mkdir -p /user/oozie"

su - hdfs -c "hdfs dfs -copyFromLocal /usr/hdp/current/oozie/
share /user/oozie/."
```

You can expect to see complaints that some files already exist. Delete any existing `/oozie/share` and replace it with the newly-extracted files.

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

5. 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', '2.5');
```

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

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

7. As root, prepare the Oozie WAR file.

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

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

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

8. Add the following property to 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.

9. Start Oozie as the Oozie user:

```
su - oozie -c "cd /grid/0/var/log/oozie; /usr/hdp/current/oozie/
oozie-server/bin/catalina.sh /usr/hdp/current/oozie/oozie-
```

```
server/bin/setclasspath.sh /usr/hdp/current/oozie-server/bin/
oozied.sh start"
```

10. Check processes.

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

3.16. Configure and Start Apache WebHCat (Templeton)

Note

The su commands in this section use keywords to represent the Service user. For example, "webhcat" is used to represent the Apache WebHCat Service user. If you are using another name for your Service users, you will need to substitute your Service user name 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 WebHCat configuration files.
   - Upload Pig, Hive and Sqoop tarballs to HDFS as the $HDFS_User. In this example, hdfs:

```
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/pig/
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/hive/
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/pig/pig.tar.gz /hdp/apps/2.3.0.
0-<$version>/pig/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hive/hive.tar.gz /hdp/apps/2.3.
0.0-<$version>/hive/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/sqoop.tar.gz /hdp/apps/2.
3.0.0-<$version>/sqoop/
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/pig
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/pig/pig.tar.gz
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/hive
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/hive/hive.tar.gz
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.tar.gz
hdfs dfs -chown -R hdfs:hadoop /hdp
```

   - 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}.

- 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>
```

- 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 2.3 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 that are allowed to submit requests by 'hcat'.

3. Start WebHCat:

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

4. Smoke test WebHCat.

- At the WebHCat host machine, run the following command:

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

- If you are using a secure cluster, run the following command:

```
curl --negotiate -u: http://cluster.$PRINCIPAL.$REALM:50111/
templeton/v1/status {"status":"ok","version":"v1"}
[machine@acme]$
```

3.17. Configure and Start Apache Pig

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

2. To validate the Apache Pig upgrade, complete the following steps:

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

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

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

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

c. Run the Pig script:

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

3.18. 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 will 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. Upload the Apache Sqoop tarball to HDFS. As the <HDFS_User>, for example 'hdfs':

   ```
   su - hdfs -c "hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/
   sqoop"

   su - hdfs -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<
   $version>/sqoop"

   su - hdfs -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.3.0.0-<
   $version>/sqoop"

   su - hdfs -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/
   sqoop.tar.gz /hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.tar.gz"

   su - hdfs -c "hdfs dfs -chmod 444 /hdp/apps/2.3.0.0-<$version>/
   sqoop/sqoop.tar.gz" hdfs
   ```

3. To validate the Sqoop upgrade, run the following command. You should see the Sqoop version information displayed.

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

   Because Sqoop is a client tool with no server component, you will need to run your own jobs to further validate the upgrade.

3.19. Configure, Start, and Validate Apache Flume

1. If you have not already done so, upgrade Apache Flume. On the Flume host machine, run the following command:

   - For **RHEL/CentOS/Oracle Linux**:

     ```
     yum upgrade flume
     ```

   - For **SLES**:

     ```
     zypper update flume

     zypper remove flume

     zypper se -s flume
     ```

     You should see Flume in the output.

     Install Flume:

     ```
     zypper install flume
     ```

2. To confirm that Flume is working correctly, create an example configuration file. The following snippet is a sample configuration that can be set using the properties file. For more detailed information, see the "Flume User Guide."

```
agent.sources = pstream
agent.channels = memoryChannel
agent.channels.memoryChannel.type = memory

agent.sources.pstream.channels = memoryChannel
agent.sources.pstream.type = exec
agent.sources.pstream.command = tail -f /etc/passwd

agent.sinks = hdfsSink
agent.sinks.hdfsSink.type = hdfs
agent.sinks.hdfsSink.channel = memoryChannel
agent.sinks.hdfsSink.hdfs.path = hdfs://tmp/flumetest
agent.sinks.hdfsSink.hdfs.fileType = SequenceFile
agent.sinks.hdfsSink.hdfs.writeFormat = Text
```

   The source here is defined as an exec source. The agent runs a given command on startup, which streams data to stdout, where the source gets it. The channel is defined as an in-memory channel and the sink is an HDFS sink.

3. Given this configuration, you can start Flume as follows:

```
$ bin/flume-ng agent --conf ./conf --conf-file example.conf --name a1 -
Dflume.root.logger=INFO,console
```

Note

The directory specified for `--conf agrument` would include a shell script flume-env.sh and potentially a log4j properties file. In this example, we pass a Java option to force Flume to log to the console and we go without a custom environment script.

4. After validating data in `hdfs://tmp/flumetest`, stop Flume and resore any backup files. Copy `/etc/flume/conf` to the conf directory in Flume hosts.

3.20. Configure, Start, and Validate Apache Mahout

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. Enter the mahout command to convert the plain text file sample-test.txt into a sequence file stored in the output directory mahouttest:

```
mahout seqdirectory --input /user/testuser/testdata --output /user/testuser/
mahouttest -ow --charset utf-8
```

3.21. Configure and Start Hue

Note

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

For HDP 2.3, use the Hue version shipped with HDP 2.3. If you have a previous version of Hue, use the following steps to upgrade Hue.

1. Migrate the 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 the database.

   ```
   cd /usr/lib/hue

   source ./build/env/bin/activate

   hue syncdb
   ```

```
deactivate
```

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

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

3.22. Finalize the Upgrade

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

1. Verify your filesystem health before finalizing the upgrade and removing the old config.
2. As the $HDFS_USER, enter:

```
hdfs dfsadmin -finalizeUpgrade
```

3.23. Install New HDP 2.3 Services

Install new HDP 2.3.0 Services (see the Non-Ambari Cluster Installation Guide):

- Atlas – a low-level service, similar to YARN, that provides metadata services to the HDP platform.
- SmartSense – a next generation subscription model that features upgrade and configuration recommendations.

4. Upgrade from HDP 1.3 to HDP 2.3 Manually

HDP 2.2 supports side-by-side installation of HDP 2.2 and above releases, which lets you perform rolling upgrades on your cluster and improve execution times for in-place upgrade. To support side-by-side installation, the HDP package version naming convention for both RPMs and Debs has changed to include the HDP 2.2 product version. For example, hadoop-hdfs is now hadoop-2.3.0.0-hdfs. HDP 2.2 marks the first release where HDP rpms, debs, and directories contain versions in the names to permit side-by-side installations of later HDP releases.

To select from the releases you have installed side-by-side, Hortonworks provides hdp-select, a command that lets you select the active verison of HDP from the versions you have selected.

The HDP packages for a complete installation of HDP 2.3 will take about 2.5 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.

Note

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

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

1. Download HDP 2.3
2. Getting Ready to Upgrade
3. Upgrade Hadoop
4. Migrate the HDP Configurations
5. Create Local Directories
6. Start HDFS
7. Upgrade Apache ZooKeeper
8. Upgrade Apache HBase
9. Upgrade Apache Hive and HCatalog
10. Upgrade Apache Oozie
11. Upgrade Apache WebHCat (Templeton)

12. Upgrade Apache Pig
13. Upgrade Apache Sqoop
14. Upgrade Apache Flume
15. Upgrade Apache Mahout
16. Upgrade Hue
17. Finalize the Upgrade
18. Install new HDP 2.3 Services

4.1. Getting Ready to Upgrade

HDP Stack upgrade involves removing HDP 1.x MapReduce and replacing it with HDP 2.x YARN and MapReduce2. Before you begin, review the upgrade process and complete the backup steps.

Note

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

1. 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 2.3 will take up about 2.5 GB of disk space.

2. Back up the following HDP 1.x directories:

   - `/etc/hadoop/conf`
   - `/etc/hbase/conf`
   - `/etc/hcatalog/conf` (Note: With HDP 2.3, `/etc/hcatalog/conf` is divided into `/etc/hive-hcatalog/conf` and `/etc/hive-webhcat/conf`. You cannot use `/etc/hcatalog/conf` in HDP 2.3.)
   - `/etc/hive/conf`
   - `/etc/pig/conf`
   - `/etc/sqoop/conf`
   - `/etc/flume/conf`
   - `/etc/mahout/conf`
   - `/etc/oozie/conf`
   - `/etc/hue/conf`
   - `/etc/zookeeper/conf`

3. **Optional:** Back up your userlogs directories, ${mapred.local.dir}/userlogs.

   Run the fsck command as the HDFS Service user and fix any errors. (The resulting file contains a complete block map of the file system.) For example, in a situation where clusters are unsecure and Kerberos credentials are not required for the HDFS user:

   ```
   su -l <HDFS_USER>
   ```

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

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

4. As the user running the HDFS service (by default, the user is hdfs), run the following commands:

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

     ```
     su -l <HDFS_USER>
     hadoop dfs -lsr / > dfs-old-lsr-1.log
     ```

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

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

     ```
     su -l <HDFS_USER>
     hadoop dfsadmin -report > dfs-old-report-1.log
     ```

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

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

   - **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.

5. HBase 0.96.0 and subsequent releases discontinue support for the HFileV1 file format, a common format prior to HBase 0.94. Before you upgrade, check for V1-format files as follows:

   - Download the Apache 0.94.24+HBase tarball in a machine. Run the binaries.

   - On the machine running the HBase 0.94 binaries, point the hbase-site.xml configuration file to a 0.94 cluster.

   - Check for HFiles in V1 format as follows:

     ```
     ./bin/hbase org.apache.hadoop.hbase.util.HFileV1Detector -p
     <hbase root data path>
     ```

   When you run the upgrade check, if “Count of HFileV1” returns any files, start the HBase shell to use major compaction for regions that have HFileV1 format. For example, the following sample output indicates that you need to compact two regions, fa02dac1f38d03577bd0f7e666f12812 and ecdd3eaee2d2fcf8184ac025555bb2af:

   ```
   Tables Processed:
   ```

```
hdfs://localhost:41020/myHBase/.META.
hdfs://localhost:41020/myHBase/usertable
hdfs://localhost:41020/myHBase/TestTable
hdfs://localhost:41020/myHBase/tCount of HFileV1: 2 HFileV1:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812/
 family/249450144068442524
hdfs://localhost:41020/myHBase/usertable/ecdd3eaee2d2fcf8184ac025555bb2af/
 family/249450144068442512

Count of corrupted files: 1
Corrupted Files:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812/
 family/1
Count of Regions with HFileV1: 2
Regions to Major Compact:
hdfs://localhost:41020/myHBase/usertable/fa02dac1f38d03577bd0f7e666f12812
hdfs://localhost:41020/myHBase/usertable/ecdd3eaee2d2fcf8184ac025555bb2af
```

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

   ```
   flush '_acl_'
   ```

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

   - Stop Nagios. On the Nagios host machine, run the following command:

     ```
     service nagios stop
     ```

   - Stop Ganglia.

     - Run this command on the Ganglia server host machine:

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

     - Run this command on all the nodes in your Hadoop cluster:

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

   - Stop Oozie. On the Oozie server host machine, run the following command:

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

     where:

     $OOZIE_USER is the Oozie Service user. For example, oozie

     $OOZIE_LOG_DIR is the directory where Oozie log files are stored (for example: /var/log/oozie).

   - Stop WebHCat. On the WebHCat host machine, run the following command:

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

     where $WEBHCAT_USER is the WebHCat Service user. For example, hcat.

- Stop Hive. On the Hive Metastore host machine and Hive Server2 host machine, run the following command:

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

  This stops the Hive Metastore and HCatalog services.

- Stop HBase.

  - Run these commands on all RegionServers:

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

  - Run these commands on the HBase Master host machine:

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

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

- Stop ZooKeeper. On the ZooKeeper host machine, run the following command:

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

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

- Stop MapReduce

  - Run these commands on all TaskTrackers slaves:

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

  - Run these commands on the HistoryServer host machine:

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

  - Run these commands on the node running the JobTracker host machine:

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

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

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

   ```
   su -l <HDFS_USER>

   hadoop dfsadmin -safemode enter
   ```

```
hadoop dfsadmin -saveNamespace
```

9. Backup your NameNode metadata.

   - Copy the following checkpoint files into a backup directory. If these directories do not exist, create them.
     - dfs.name.dir/edits
     - dfs.name.dir/image/fsimage
     - dfs.name.dir/current/fsimage
   - Store the layoutVersion of the namenode.

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

10. If you have a prior HDFS upgrade in progress, finalize it if you have not done so already.

    ```
    su -l <HDFS_USER>

    hadoop dfsadmin -finalizeUpgrade
    ```

11. **Optional:** Back up the Hive Metastore database.

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

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

Table 4.1. Hive Metastore Database Backup and Restore

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

12. **Optional**: Back up the Oozie Metastore database.

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

Table 4.2. Oozie Metastore Database Backup and Restore

Database Type	Backup	Restore

MySQL	`mysqldump $dbname > $outputfilename.sql`
	
	For example:
	
	`mysqldump oozie > / tmp/ mydir/backup_oozie.sql`
		`mysql $dbname < $inputfilename.sql`
		
		For example:
		
		`mysql oozie </tmp/mydir/ backup_oozie.sql`

Postgres	`sudo -u $username pg_dump $databasename > $outputfilename.sql`
	
	For example:
	
	`sudo -u postgres pg_dump oozie > /tmp/mydir/ backup_oozie.sql`
		`sudo -u $username psql $databasename < $inputfilename.sql`
		
		For example:
		
		`sudo -u postgres psql oozie </tmp/mydir/ backup_oozie.sql`

13. **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 4.3. 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	/etc/init.d/hue stop cd /var/lib/hue

Database Type	Backup	Restore
	mkdir ~/hue_backup sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak /etc/init.d/hue start	mv desktop.db desktop.db.old sqlite3 desktop.db < ~/hue_backup/desktop.bak /etc/init.d/hue start

14. Stop HDFS.

    a. Run these commands on all DataNodes:

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

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

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

    c. Run these commands on the NameNode host machine:

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

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

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

    a. Run the following command:

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

    b. Verify 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.

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 will print 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, a user would specify:

```
-upgrade -renameReserved .snapshot=.my-snapshot,.reserved=.my-
reserved.
```

If no key-value pairs are specified with -renameReserved, the NameNode will then suffix 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` will rename all applicable existing files in the cluster. This may impact cluster applications.

4.2. Upgrade HDP 1.3 Components

The upgrade process to HDP 2.3 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 1.3 components. This command uninstalls the HDP 1.3 components. It leaves the user data, metadata and modified configurations in place and does not delete them:

   ```
   yum erase "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
   "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*"
   "hive*" "flume*" "mahout*" "hue" "hue-common" "hue-shell"
   "hdp_mon_nagios_addons"
   ```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos6/2.x/
     updates/2.3.0.0/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-2.3.0.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.3.0.0 Hortonworks Data Platform Version - HDP-2.3.0.0
     ```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 1.3 components:

   ```
   yum install "hadoop" "hadooplzo" "webhcat" "oozie" "collectd"
   "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "flume"
   "mahout" "hue" "knox" "hdp_mon_nagios_addons"
   ```

6. 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 1.3 components. This command uninstalls the HDP 1.3 components. It leaves the user data and metadata but deletes your modified configurations:

   ```
   yum erase "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
   "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*"
   "hive*" "flume*" "mahout*" "hue" "hue-common" "hue-shell"
   "hdp_mon_nagios_addons"
   ```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

     ```
     wget -nv http://public-repo-1.hortonworks.com/HDP/centos5/2.x/
     updates/2.3.0.0/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-2.3.0.0 directory:

     ```
     Loaded plugins: fastestmirror, security
     Loading mirror speeds from cached hostfile
     * base: mirrors.cat.pdx.edu
     * extras: linux.mirrors.es.net
     * updates: mirrors.usc.edu
     repo id repo namestatus
     HDP-2.3.0.0 Hortonworks Data Platform Version - HDP-2.3.0.0
     ```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 1.3 components:

   ```
   yum install "hadoop" "hadooplzo" "webhcat" "oozie" "collectd"
   "gccxml" "pig" "sqoop" "zookeeper" "hbase" "hive" "flume"
   "mahout" "hue" "knox" "hdp_mon_nagios_addons"
   ```

6. 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 SP 1

1. On all hosts, clean the yum repository.

   ```
   zypper clean -all
   ```

2. Remove your old HDP 1.3 components. This command uninstalls the HDP 1.3 components. It leaves the user data and metadata but deletes your modified configurations:

   ```
   zypper rm "hadoop*" "webhcat*" "hcatalog*" "oozie*" "collectd*"
   "gccxml*" "pig*" "hdfs*" "sqoop*" "zookeeper*" "hbase*"
   "hive*" "flume*" "mahout*" "hue" "hue-common" "hue-shell"
   "hdp_mon_nagios_addons"
   ```

3. Remove your old hdp.repo file:

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

4. Install the HDP 2.3 repo:

   - Download the hdp.repo file:

```
wget -nv http://public-repo-1.hortonworks.com/HDP/
sles11sp1/2.x/updates/2.3.0.0/hdp.repo -O /etc/zypp/repos.d/
hdp.repo
```

5. Install the HDP 2.3 versions of the components that you want to upgrade. For example, if you installed and want to upgrade all HDP 1.3 components:

```
zypper install "hadoop" "webhcat" "oozie" "collectd" "gccxml"
"pig" "hdfs" "sqoop"

"zookeeper" "hbase" "hive" "flume" "mahout" "hue"
"hdp_mon_nagios_addons"

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

zypper up -r HDP-2.3.0.0

zypper install oozie-client
```

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

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

   No component files names should appear in the returned list.

4.3. Symlink Directories with hdp-select

Warning

HDP 2.3 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.

- Before you run hdp-select, you must remove one link:

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

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

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

  For example:

```
/usr/bin/hdp-select set all 2.3.0.0-2
```

4.4. Configure and Start Apache ZooKeeper

Tip

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

RHEL/CentOS/Oracle Linux

1. Replace your configuration after upgrading ZooKeeper. Replace the ZooKeeper template configuration in `/etc/zookeeper/conf`.
2. Start ZooKeeper.

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

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

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

SLES

1. Replace your configuration after upgrading. Replace the ZooKeeper template configuration in /etc/zookeeper/conf.
2. Start ZooKeeper.

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

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

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

4.5. Configure and Start 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 . make sure you remove lines 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 will 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 in HDP 2.3. make sure you remove lines 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 will not load as this is not where lzo is installed.

4.6. Migrate the HDP Configurations

Configurations and configuration file names have changed between HDP 1.3.2 (Hadoop 1.2.x) and HDP 2.3 (Hadoop 2.6). To upgrade to HDP 2.3.x, back up your current configuration files, download the new HDP 2.3 files, and compare. The following tables provide mapping information to make the comparison between releases easier.

To migrate the HDP Configurations

1. Edit `/etc/hadoop/conf/core-site.xml` and set hadoop.rpc.protection from none to authentication.

   

   Note

   Hadoop lets cluster administrators control the quality of protection in the configuration parameter “hadoop.rpc.protection” in core-site.xml. It is an optional parameter in HDP 2.3. If not present, the default QOP setting of “auth” is used, which implies “authentication only”.

   Valid values for this parameter are: “authentication” : Corresponds to “auth” “integrity” : Corresponds to “auth-int” “privacy” : Corresponds to “auth-conf”

   The default setting is authentication-only because integrity checks and encryption are a performance cost.

2. Copy your /etc/hcatalog/conf configurations to /etc/hive-hcatalog/conf and /etc/hive-webhcat as appropriate.

3. Copy log4j.properties from the hadoop config directory of the companion files to /etc/hadoop/conf. The file should have owners and permissions similar to other files in /etc/hadoop/conf.

4. Download the your HDP 2.3.x companion files from Download Companion Files and migrate your HDP 1.x configuration.

5. Copy these configurations to all nodes in your clusters.

   - /etc/hadoop/conf
   - /etc/hbase/conf
   - /etc/hcatalog/conf
   - /etc/hive/conf
   - /etc/pig/conf
   - /etc/sqoop/conf
   - /etc/flume/conf
   - /etc/mahout/conf
   - /etc/oozie/conf
   - /etc/zookeeper/conf

   

   Note

   Upgrading the repo using `yum` or `zypper` resets all configurations. Prepare to replace these configuration directories each time you perform a yum or zypper rmgrade.

6. Review the following HDP 1.3.2 Hadoop Core configurations and the new configurations or locations in HDP 2.3.x.

Table 4.4. HDP 1.3.2 Hadoop Core Site (core-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
`fs.default.name`	core-site.xml	`fs.defaultFS`	core-site.xml
`fs.checkpoint.dir`	core-site.xml	`dfs.namenode.checkpoint.dir`	hdfs-site.xml
`fs.checkpoint.edits.dir`	core-site.xml	`dfs.namenode.checkpoint.edits.dir`	hdfs-site.xml
`fs.checkpoint.period`	core-site.xml	`dfs.namenode.checkpoint.period`	hdfs-site.xml
`io.bytes.per.checksum`	core-site.xml	`dfs.bytes-per-checksum`	hdfs-site.xml
`dfs.df.interval`	hdfs-site	`fs.df.interval`	core-site.xml
`hadoop.native.lib`	core-site.xml	`io.native.lib.available`	core-site.xml
`hadoop.configured.node.mapping`	–	`net.topology.configured.node.mapping`	core-site.xml
`topology.node.switch.mapping.impl`	core-site-xml	`net.topology.node.switch.mapping.impl`	core-site.xml
`topology-script.file.name`	core-site.xml	`net.topology.script.file.name`	core-site.xml

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
topology.script.number.args	core-site.xml	net.topology.script.number.args	core-site.xml

Note

The hadoop.rpc.protection configuration property in core- site.xml needs to specify authentication, integrity and/or privacy. No value defaults to authentication, but an invalid value such as "none" causes an error.

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

Table 4.5. HDP 1.3.2 Hadoop Core Site (hdfs-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
dfs.block.size	hdfs-site.xml	dfs.blocksize	hdfs-site.xml
dfs.write.packet.size	hdfs-site.xml	dfs.client-write-packet-size	hdfs-site.xml
dfs.https.client.keystore.resource	hdfs-site.xml	dfs.client.https.keystore.resource	hdfs-site.xml
dfs.https.need.client.auth	hdfs-site.xml	dfs.client.https.need-auth	hdfs-site.xml
dfs.read.prefetch.size	hdfs-site.xml	dfs.bytes-per-checksum	hdfs-site.xml
dfs.socket.timeout	hdfs-site.xml	dfs.client.socket-timeout	hdfs-site.xml
dfs.balance.bandwidthPerSec	hdfs-site.xml	dfs.datanode.balance.bandwidthPerSec	hdfs-site.xml
dfs.data.dir	hdfs-site.xml	dfs.datanode.data.dir	hdfs-site.xml
dfs.datanode.max.xcievers	hdfs-site.xml	dfs.datanode.max.transfer.threads	hdfs-site.xml
session.id	hdfs-site.xml	dfs.metrics.session-id	hdfs-site.xml
dfs.access.time.precision	hdfs-site.xml	dfs.namenode.accesstime.precision	hdfs-site.xml
dfs.backup.address	hdfs-site.xml	dfs.namenode.backup.address	hdfs-site.xml
dfs.backup.http.address	hdfs-site.xml	dfs.namenode.backup.http-address	hdfs-site.xml
fs.checkpoint.dir	hdfs-site.xml	dfs.namenode.checkpoint.dir	hdfs-site.xml
fs.checkpoint.edits.dir	hdfs-site.xml	dfs.namenode.checkpoint.edits.dir	hdfs-site.xml
fs.checkpoint.period	hdfs-site.xml	dfs.namenode.checkpoint.period	hdfs-site.xml
dfs.name.edits.dir	hdfs-site.xml	dfs.namenode.edits.dir	hdfs-site.xml
heartbeat.recheck.interval	hdfs-site.xml	dfs.namenode.heartbeat.recheck-interval	hdfs-site.xml
dfs.http.address	hdfs-site.xml	dfs.namenode.http-address	hdfs-site.xml

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
dfs.https.address	hdfs-site.xml	dfs.namenode.https-address	hdfs-site.xml
dfs.max.objects	hdfs-site.xml	dfs.namenode.max.objects	hdfs-site.xml
dfs.name.dir	hdfs-site.xml	dfs.namenode.name.dir	hdfs-site.xml
dfs.name.dir.restore	hdfs-site.xml	dfs.namenode.name.dir.restore	hdfs-site.xml
dfs.replication.considerLoad	hdfs-site.xml	dfs.namenode.replication.considerLoad	hdfs-site.xml
dfs.replication.interval	hdfs-site.xml	dfs.namenode.replication.interval	hdfs-site.xml
dfs.max-repl-streams	hdfs-site.xml	dfs.namenode.replication.max-streams	hdfs-site.xml
dfs.replication.min	hdfs-site.xml	dfs.namenode.replication.min	hdfs-site.xml
dfs.replication.pending.timeout.sec	hdfs-site.xml	dfs.namenode.replication.pending.timeout-sec	hdfs-site.xml
dfs.safemode.extension	hdfs-site.xml	dfs.namenode.safemode.extension	hdfs-site.xml
dfs.safemode.threshold.pct	hdfs-site.xml	dfs.namenode.secondary.threshold-pct	
dfs.secondary.http.address	hdfs-site.xml	dfs.namenode.secondary.http-address	hdfs-site.xml
dfs.permissions	hdfs-site.xml	dfs.permissions.enabled	hdfs-site.xml
dfs.permissions.supergroup	hdfs-site.xml	dfs.permissions.superusergroup	hdfs-site.xml
dfs.df.interval	hdfs-site.xml	fs.df.interval	core-site.xml
dfs.umaskmode	hdfs-site.xml	fs.permissions.umask-mode	hdfs-site.xml

8. Review the following HDP 1.3.2 MapReduce Configs and their new HDP 2.x mappings.

Table 4.6. HDP 1.3.2 Configs now in Capacity Scheduler for HDP 2.x (mapred-site.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
mapred.map.child.java.opts	mapred-site.xml	mapreduce.map.java.opts	mapred-site.xml
mapred.job.map.memory.mb	mapred-site.xml	mapred.job.map.memory.mb	mapred-site.xml
mapred.reduce.child.java.opts	mapred-site.xml	mapreduce.reduce.java.opts	mapred-site.xml
mapreduce.job.reduce.memory.mb	mapred-site.xml	mapreduce.reduce.memory.mb	mapred-site.xml
security.task.umbilical.protocol.acl	mapred-site.xml	security.job.task.protocol.acl	mapred-site.xml

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

Table 4.7. HDP 1.3.2 Configs now in capacity scheduler for HDP 2.x (capacity-scheduler.xml)

HDP 1.3.2 config	HDP 1.3.2 config file	HDP 2.3 config	HDP 2.3 config file
mapred.queue.names	mapred-site.xml	yarn.scheduler.capacity.root.queues	capacity-scheduler.xml
mapred.queue.default.acl-submit.job	mapred-queue-acls.xml	yarn.scheduler.capacity.root.default.acl_submit_jobs	capacity-scheduler.xml
mapred.queue.default.acl.administer-jobs	mapred-queue-acls.xml	yarn.scheduler.capacity.root.default acl_administer_jobs	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.capacity	capacity-scheduler.xml	yarn-scheduler.capacity.root.default.capacity	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.user-limit-factor	capacity-scheduler.xml	yarn.scheduler.capacity.root.default user-limit-factor	capacity-scheduler.xml
mapred.capacity-scheduler.queue.default.maximum-capacity	capacity-scheduler.xml	yarn.scheduler.capacity.root.default maximum-capacity	capacity-scheduler.xml
mapred.queue.default.state	capacity-scheduler.xml	yarn.scheduler.capacity.root.default.state	capacity-scheduler.xml

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

    Paths have changed in HDP 2.3 to `/usr/hdp/current`. You must remove lines such as:

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

Table 4.8. HDP 1.3.2 Configs and HDP 2.x for hadoop-env.sh

HDP 1.3.2 config	HDP 2.3 config	Description
JAVA_HOME	JAVA_HOME	Java implementation to use
HADOOP_HOME_WARN_SUPPRESS	HADOOP_HOME_WARN_SUPPRESS	–
HADOOP_CONF_DIR	HADOOP_CONF_DIR	Hadoop configuration directory
not in hadoop-env.sh.	HADOOP_HOME	–
not in hadoop-env.sh.	HADOOP_LIBEXEC_DIR	–
HADOOP_NAMENODE_INIT_HEAPSIZE	HADOOP_NAMENODE_INIT_HEAPSIZE	–
HADOOP_OPTS	HADOOP_OPTS	Extra Java runtime options; empty by default
HADOOP_NAMENODE_OPTS	HADOOP_NAMENODE_OPTS	Command-specific options appended to HADOOP-OPTS

HDP 1.3.2 config	HDP 2.3 config	Description
HADOOP_JOBTRACKER_OPTS	not in hadoop-env.sh.	Command-specific options appended to HADOOP-OPTS
HADOOP_TASKTRACKER_OPTS	not in hadoop-env.sh.	Command-specific options appended to HADOOP-OPTS
HADOOP_DATANODE_OPTS	HADOOP_DATANODE_OPTS	Command-specific options appended to HADOOP-OPTS
HADOOP_BALANCER_OPTS	HADOOP_BALANCER_OPTS	Command-specific options appended to HADOOP-OPTS
HADOOP_SECONDARYNAMENODE_OPTS	HADOOP_SECONDARYNAMENODE_OPTS	Command-specific options appended to HADOOP-OPTS
HADOOP_CLIENT_OPTS	HADOOP_CLIENT_OPTS	Applies to multiple commands (fs, dfs, fsck, distcp, etc.)
HADOOP_SECURE_DN_USER	not in hadoop-env.sh.	Secure datanodes, user to run the datanode as
HADOOP_SSH_OPTS	HADOOP_SSH_OPTS	Extra ssh options.
HADOOP_LOG_DIR	HADOOP_LOG_DIR	Directory where log files are stored in the secure data environment.
HADOOP_SECURE_DN_LOG_DIR	HADOOP_SECURE_DN_LOG_DIR	Directory where pid files are stored; /tmp by default.
HADOOP_PID_DIR	HADOOP_PID_DIR	Directory where pid files are stored, /tmp by default.
HADOOP_SECURE_DN_PID_DIR	HADOOP_SECURE_DN_PID_DIR	Directory where pid files are stored, /tmp by default.
HADOOP_IDENT_STRING	HADOOP_IDENT_STRING	String representing this instance of hadoop. $USER by default
not in hadoop-env.sh.	HADOOP_MAPRED_LOG_DIR	–
not in hadoop-env.sh.	HADOOP_MAPRED_PID_DIR	–
not in hadoop-env.sh.	JAVA_LIBRARY_PATH	–
not in hadoop-env.sh.	JSVC_HOME	For starting the datanode on a secure cluster

Note

Some of the configuration settings refer to the variable HADOOP_HOME. The value of HADOOP_HOME is automatically inferred from the location of the startup scripts. HADOOP_HOME is the parent directory of the bin directory that holds the Hadoop scripts. In many instances this is $HADOOP_INSTALL/hadoop.

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

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

<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>$resourcemanager.full.hostname:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>
```

```
<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>$resourcemanager.full.hostname:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.address</name>
 <value>$resourcemanager.full.hostname:8050</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>$resourcemanager.full.hostname:8141</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

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

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

<property>
 <name>yarn.log.server.url</name>
 <value>http://$jobhistoryserver.full.hostname:19888/jobhistory/logs/</
 value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>$resourcemanager.full.hostname:8088</value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.nodemanager.admin-env</name>
 <value>MALLOC_ARENA_MAX=$MALLOC_ARENA_MAX</value>
 <description>Restrict the number of memory arenas to prevent
    excessive VMEM use by the glib arena allocator.
    For example, MALLOC_ARENA_MAX=4</description>
</property>
```

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

```
<property>
 <name>yarn.resourcemanager.scheduler.class</name>
```

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

<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>$resourcemanager.full.hostname:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>$resourcemanager.full.hostname:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.address</name>
 <value>$resourcemanager.full.hostname:8050
 </value><description>Enter your ResourceManager hostname.
 </description></property>

<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>$resourcemanager.full.hostname:8141</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

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

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

<property>
 <name>yarn.log.server.url</name>
 <value>http://$jobhistoryserver.full.hostname:19888/jobhistory/logs/</
 value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>$resourcemanager.full.hostname:8088</value>
 <description>URL for job history server</description>
</property>

<property>
```

```
 <name>yarn.nodemanager.admin-env</name>
 <value>MALLOC_ARENA_MAX=$MALLOC_ARENA_MAX</value>
 <description>Restrict the number of memory arenas to prevent excessive VMEM
 use by
 the glib arena allocator.For example, MALLOC_ARENA_MAX=4</description>
</property>
```

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

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

<property>
 <name>yarn.resourcemanager.resource-tracker.address</name>
 <value>$resourcemanager.full.hostname:8025</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.scheduler.address</name>
 <value>$resourcemanager.full.hostname:8030</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.address</name>
 <value>$resourcemanager.full.hostname:8050</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

<property>
 <name>yarn.resourcemanager.admin.address</name>
 <value>$resourcemanager.full.hostname:8141</value>
 <description>Enter your ResourceManager hostname.</description>
</property>

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

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

<property>
 <name>yarn.log.server.url</name>
 <value>http://$jobhistoryserver.full.hostname:19888/jobhistory/logs/
```

```
 </value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.resourcemanager.webapp.address</name>
 <value>$resourcemanager.full.hostname:8088</value>
 <description>URL for job history server</description>
</property>

<property>
 <name>yarn.nodemanager.admin-env</name>
 <value>MALLOC_ARENA_MAX=$MALLOC_ARENA_MAX</value>
 <description>Restrict the number of memory arenas to prevent excessive
 VMEM use by the glib arena allocator. For example,
 MALLOC_ARENA_MAX=4</description>
</property>
```

14. Adding the following properties to the mapred-site.xml file:

```
<property>
 <name>mapreduce.jobhistory.address</name>
 <value>$jobhistoryserver.full.hostname:10020</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>

<property>
 <name>mapreduce.jobhistory.webapp.address</name>
 <value>$jobhistoryserver.full.hostname:19888</value>
 <description>Enter your JobHistoryServer hostname.</description>
</property>

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

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

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

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

</property>
 <name>mapreduce.jobhistory.keytab</name>
 <value>/etc/security/keytabs/jhs.service.keytab</value>
 <description>Kerberos keytab file for the MapReduce JobHistory Server.</
description>
</property>
```

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

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

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

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

18. Replace the default memory configuration settings in yarn-site.xml and mapred-site.xml with the YARN and MapReduce memory configuration settings you calculated previously.

4.7. Create Local Directories

You must create local directories for YARN on each NodeManager host in your cluster (in HDP-2, the NodeManager replaces the TaskTracker) and set the appropriate permissions for the YARN log directories.

1. Set the permissions in the yarn.nodemanager.local-dirs directories. Run these commands on all DataNodes in your cluster.

   ```
   chown -R yarn:hadoop ${yarn.nodemanager.local-dirs}

   chmod 755 ${yarn.nodemanager.local-dirs}
   ```

   where ${yarn.nodemanager.local-dirs} is your local directory.

2. Change the permissions of the directories in yarn.nodemanager.log-dirs. If these directories do not exist, you can create them using the instructions in the Create Directories section of the Non-Ambari Cluster Installation Guide. Run these commands on all DataNodes in your cluster.

   ```
   chown -R yarn:hadoop ${yarn.nodemanager.log-dirs}

   chmod 755 ${yarn.nodemanager.log-dirs}
   ```

   where ${yarn.nodemanager.log-dirs} is your log directory.

3. Create directories for YARN_LOG_DIR and YARN_PID_DIR.

   - Open /etc/hadoop/conf/yarn-env.sh
   - Write down your values for YARN_LOG_DIR and YARN_PID_DIR, as the following instructions require values for the ${YARN_LOG_DIR} and ${YARN_PID_DIR}.

   For example, in yarn-env.sh:

   ```
   YARN_LOG_DIR=/grid/0/var/log/hadoop/yarn

   YARN_PID_DIR=/grid/0/var/run/hadoop/yarn
   ```

4. Make directories for ${YARN_LOG_DIR} and ${YARN_PID_DIR} and set the appropriate permissions for them.

   ```
   mkdir ${YARN_LOG_DIR}

   chown yarn:hadoop ${YARN_LOG_DIR}

   chmod 755 ${YARN_LOG_DIR}

   mkdir ${YARN_PID_DIR}

   chown yarn:hadoop ${YARN_PID_DIR}

   chmod 755 ${YARN_PID_DIR}
   ```

4.8. Start Hadoop Core

Warning

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

Start HDFS, executing commands as $HDFS_USER.

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 will not enter the standby state as usual. Rather, this NameNode will immediately enter 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 will be 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 will download 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.

4.9. Verify HDFS Filesystem Health

Analyze if the filesystem is healthy.

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

   ```
   su - $HDFS_USER
   ```

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

2. Run hdfs namespace and report.

   a. List directories.

      ```
      su -l <HDFS_USER>
      ```

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

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

      ```
      su -l <HDFS_USER>
      ```

      ```
      hdfs dfsadmin -report > dfs-new-report-1.log
      ```

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

```
su - $HDFS_USER

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.

d. From the Namenode WebUI, see if all DataNodes are up and running.

```
http://<namenode>:50070
```

4.10. Configure YARN and MapReduce

After you install Hadoop, modify your configs.

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

```
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/mapreduce/

hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hadoop/
mapreduce.tar.gz /hdp/apps/2.3.0.0-<$version>/mapreduce/

hdfs dfs -chown -R hdfs:hadoop /hdp

hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/mapreduce

hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/mapreduce/
mapreduce.tar.gz
```

2. Make the following changes to mapred-site.xml:

   - Add the following property:

```
<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}.

3. Add the following property to 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. For secure clusters, you must create and configure the container-executor.cfg configuration file:

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

```
yarn.nodemanager.linux-container-executor.group=hadoop
```

```
banned.users=hdfs,yarn,mapred
min.user.id=1000
```

- 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
  ```

4.11. Start YARN/MapReduce Services

Add the following properties to yarn-site.xml to configure Application Timeline Server (ATS):

```
yarn.timeline-service.leveldb-timeline-store.path=/var/log/hadoop-yarn/
timeline
yarn.timeline-service.leveldb-timeline-store.ttl-interval-ms=300000

** If you are upgrading to HDP 2.1.3 or higher, use the following setting:
 yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.timeline.
LeveldbTimelineStore**

** If you are upgrading to HDP 2.1.2, use the following setting:
 yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.
applicationhistoryservice.timeline.LeveldbTimelineStore **

yarn.timeline-service.ttl-enable=true
yarn.timeline-service.ttl-ms=2678400000
yarn.timeline-service.generic-application-history.store-class=org.apache.
hadoop.yarn.server.applicationhistoryservice.NullApplicationHistoryStore
yarn.timeline-service.webapp.address={PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:8188
yarn.timeline-service.webapp.https.address=
{PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:8190
yarn.timeline-service.address={PUT_THE_FQDN_OF_ATS_HOST_NAME_HERE}:10200
```

```
HIVE (hive-site.xml)
hive.execution.engine=mr
hive.exec.failure.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.pre.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.tez.container.size={map-container-size}

*If mapreduce.map.memory.mb > 2GB then set it equal to mapreduce.map.memory.
 Otherwise, set it equal to mapreduce.reduce.memory.mb*

hive.tez.java.opts=-server -Xmx800m -Djava.net.preferIPv4Stack=true -
XX:NewRatio=8 -XX:+UseNUMA -XX:+UseParallelGC
```

Use configuration values appropriate for your environment. For example, the value "800" in the preceding example is an example, not a requirement.

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

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

2. Start the ResourceManager on your previous JobTracker host.

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

   ps -ef | grep -i resourcemanager
   ```

3. Prepare the NodeManager hosts.

   a. Change permissions for /usr/hdp/current/hadoop-yarn/bin/container- executor.cfg:

      ```
      chown yarn:hadoop /usr/hdp/current/hadoop-yarn/bin/container-
      executor

      chmod -R 650 /usr/hdp/current/hadoop-yarn/bin/container-
      executor
      ```

   b. On all NodeManager hosts, add the yarn user to the hadoop group.

      For example, if you are using CentOS6:

      ```
      usermod -a -G hadoop yarn
      ```

4. Start the NodeManager on your previous TaskTracker hosts.

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

```
ps -ef | grep -i jobhistoryserver
```

4.12. 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=100000000 test-after-upgrade
```

You should see messages similar to:

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

You have successfully submitted your first MapReduce job in HDP 2.x. The next steps will upgrade your other components to 2.x.

Basic troubleshooting:

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

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

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

2. Error messages. Access the ApplicationMaster WebUI to view the container logs.

   a. Looking at your console logs for MapReduce job, there is a line in this format:

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

   b. Go to the URL and select the job link.

   c. Select the logs link under ApplicationMaster table. It will redirect you to the container logs. Error messages display here.

4.13. Configure and Start Apache HBase

The hbase.bucketcache.percentage.in.combinedcache is removed in HDP-2.3.0. This simplifies the configuration of block cache. BucketCache configurations from HDP 2.1 will need to be recalculated to attain identical memory allotments in HDP-2.3.0. The L1 LruBlockCache will be whatever hfile.block.cache.size is set to and the L2 BucketCache will be whatever hbase.bucketcache.size is set to.

1. Upgrade Apache HBase.

   ```
   hbase upgrade -execute
   ```

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

3. 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"
```

4. Check processes.

```
ps -ef | grep -i hmaster

ps -ef | grep -i hregion
```

4.14. Configure and Start Apache Hive and Apache HCatalog

Warning

In HDP 2.1.3 and later, the Decimal data type is now treated as the type Decimal(10,0): 10 digits of precision and 0 scale. This is a change from the variable precision and scale available in Apache Hive 0.11.0 and Hive 0.12.0, which allowed up to 38 digits of precision and unlimited scale.

To avoid unintended rounding of decimal data, sites that were previously running Hive 0.11.0 and Hive 0.12.0 may need to migrate tables with Decimal columns after upgrading to Hive 0.13.0. For details, see the Apache Hive wiki. For assistance with upgrades that involve Decimal data, please contact Hortonworks Support.

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

```
datanucleus.autoCreateSchema=false
```

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

3. Upgrade the Hive Metastore database schema by running the upgrade scripts included in HDP for your metastore database and then running the schematool to upgrade to Hive 14:

Database	Run as user...	This metastore upgrade script
MySQL	root	>cd /usr/hdp/current/hive-metastore/scripts/metastore/upgrade/mysql > mysql hivemysql > source upgrade-0.11.0-to-0.12.0.mysql.sql > source upgrade-0.12.0-to-0.13.0.mysql.sql

Database	Run as user...	This metastore upgrade script
		> source upgrade-0.13.0-to-0.14.0.mysql.sql
Postgres	root	cd /usr/hdp/current/hive-metastore/scripts/metastore/upgrade/postgres psql -d hive -a -f upgrade-0.11.0-to-0.12.0.postgres.sql psql -d hive -a -f upgrade-0.12.0-to-0.13.0.postgres.sql psql -d hive -a -f upgrade-0.13.0-to-0.14.0.postgres.sql
Oracle	root	cd /usr/hdp/2.3.0.0-<version>/hive-metastore/scripts/metastore/upgrade/oracle sqlplus SQL> @upgrade-0.11.0-to-0.12.0.oracle.sql SQL> @upgrade-0.12.0-to-0.13.0.oracle.sql SQL> @upgrade-0.13.0-to-0.14.0.oracle.sql

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

Note

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

```
sudo <POSTGRES_USER>

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

```
<property>
 <name>hive.metadata.export.location</name>
 <value>export HIVE_AUX_JARS_PATH=/usr/hdp/2.3.0.0-<version>/hive-hcatalog/
share/hcatalog/hive-hcatalog-core.jar</value>
</property>
```

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

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

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

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

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

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

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

<property>
 <name>javax.jdo.option.ConnectionDriverName</name>
```

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

- 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>
```

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

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

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

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

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

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

  To enable HiveServer2, leave this property value empty.

```
<property>
  <name>hive.metastore.uris</name>
  <value>thrift://$metastore.server.full.hostname:9083</value>
  <description>URI for client to contact metastore server.
    To enable HiveServer2, leave the property value empty.
  </description>
</property>
```

Note

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

7. Disable autocreation of schemas to match HDP 2.1+ configurations. Edit hive-site.xml to set the value of `datanucleus.autoCreateSchema` to false.

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

8. Start Hive. On the Hive Metastore host machine, run the following commands:

```
su - hive

nohup /usr/hdp/current/hive-metastore/bin/hive --service
metastore>/var/log/hive/hive.out 2>/var/log/hive/hive.log &
```

9. Start Hive Server2.

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

```
su - hive

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

   where $HIVE_USER is the Hive Service user. For example, hive.

4.15. Configure and Start Apache Oozie

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/2.3.0.0-<version>/oozie-server/libext-customer` directory.

```
cd /usr/hdp/2.3.0.0-<version>/oozie-server

mkdir libext-customer
```

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

```
chmod -R 777 /usr/hdp/2.3.0.0-<version>/oozie-server/libext-
customer
```

3. Copy these files to the libext-customer directory

   ```
   cp /usr/hdp/2.3.0.0-<version>/hadoop-client/lib/
   hadoop*lzo*.jar /usr/hdp/current/oozie-server/libext-customer
   ```

   ```
   cp /usr/share/HDP-oozie/ext.zip /usr/hdp/2.3.0.0-<version>/
   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.

   ```
   cd /usr/hdp/2.3.0.0-<version>/oozie-server
   ```

   ```
   tar xzvf /usr/hdp/2.3.0.0-<version>/oozie-server/oozie-
   sharelib.tar.gz
   ```

   ```
   su - hdfs -c "hdfs dfs -mkdir -p /user/oozie"
   ```

   ```
   su - hdfs -c "hdfs dfs -copyFromLocal /usr/hdp/current/oozie-
   server/share /user/oozie/."
   ```

   You can expect warnings that some files already exist. Delete any existing /oozie/share and replace it with the newly-extracted files.

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

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

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', '2.5');
   ```

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. 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.

11. Start Oozie as the Oozie user:

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

12. Check processes.

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

4.16. Configure and Start Apache WebHCat (Templeton)

RHEL/CentOS/Oracle Linux

1. Copy the appropriate configurations from /etc/hcatalog/conf to /etc/hive- webhcat/conf/.

2. Copy the new Pig, Hive and Hadoop-streaming jars to HDFS using the path you specified in ./etc/hive-webhcat/conf/ and change ownership to the hcat user with 755 permissions. For example:

   ```
   hdfs dfs -copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /usr/
   share/HDP-webhcat/pig.tar.gz/usr/hdp/version/hadoop-mapreduce/
   hadoop-streaming.jar hdfs:///apps/webhcat/.

   hdfs dfs -chmod -R 755 hdfs:///apps/webhcat/*

   hdfs dfs -chown -R hcat hdfs:///apps/webhcat/*
   ```

3. Replace your WebHCat configuration after upgrading. Copy your modified /etc/webhcat/conf from the template to the configuration directory in all your WebHCat hosts.

4. Start WebHCat:

   ```
   sudo su -l $WEBHCAT_USER -c "//hive-hcatalog/sbin/
   webhcat_server.sh start"
   ```

5. Smoke test WebHCat.

   On the WebHCat host machine, run the following command:

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

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

   ```
   curl --negotiate -u:http://cluster.$PRINCIPAL.$REALM:50111/
   templeton/v1/ status{"status":"ok","version":"v1"}
   [machine@acme]$
   ```

6. Remove shared libraries from old Templeton installation.

   On the WebHCat host machine, run the following command:

   ```
   sudo su -l $HDFS_USER -c "hdfs dfs -rmr -skipTrash /apps/
   templeton" rm -rf /usr/share/HDP-templeton
   ```

   where

   - *$WEBHCAT_USER*is the WebHCat Service user. For example, hcat.
   - *$HDFS_USER*is the HDFS Service user. For example, hdfs.

SLES

1. Copy the appropriate configurations from /etc/hcatalog/conf to /etc/hive-webhcat/conf/.

2. Copy the new Pig, Hive and Hadoop-streaming jars to HDFS using the path you specified in ./etc/hive-webhcat/conf/ and change ownership to the hcat user with 755 permissions. For example:

```
hdfs dfs -copyFromLocal /usr/share/HDP-webhcat/hive.tar.gz /usr/
share/HDP-webhcat/pig.tar.gz/usr/hdp/version/hadoop-mapreduce/
hadoop-streaming.jar hdfs:///apps/webhcat/.

hdfs dfs -chmod -R 755 hdfs:///apps/webhcat/*

hdfs dfs -chown -R hcat hdfs:///apps/webhcat/*
```

3. Replace your WebHCat configuration after upgrading. Copy your modified /etc/webhcat/conf from the template to the configuration directory in all your WebHCat hosts.

4. Modify the WebHCat configuration files.

   - Upload Pig, Hive and Sqoop tarballs to HDFS as the $HDFS_User. In this example, hdfs:

```
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/pig/
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/hive/
hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/pig/pig.tar.gz /hdp/apps/2.3.0.
0-<$version>/pig/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/hive/hive.tar.gz /hdp/apps/2.3.
0.0-<$version>/hive/
hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/sqoop.tar.gz /hdp/apps/2.
3.0.0-<$version>/sqoop/
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/pig
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/pig/pig.tar.gz
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/hive
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/hive/hive.tar.gz
hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/sqoop
hdfs dfs -chmod -R 444 /hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.tar.gz
hdfs dfs -chown -R hdfs:hadoop /hdp
```

   - 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}.

- 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>
```

- 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 2.3 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'.

5. Start WebHCat:

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

6. Smoke test WebHCat.

   On the WebHCat host machine, run the following command:

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

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

   ```
   curl --negotiate -u:http://cluster.$PRINCIPAL.$REALM:50111/
   templeton/v1/ status{"status":"ok","version":"v1"}
   [machine@acme]$
   ```

7. Remove shared libraries from old Templeton installation.

   On the WebHCat host machine, run the following command:

   ```
   sudo su -l $HDFS_USER -c "hdfs dfs -rmr -skipTrash /apps/
   templeton" rm -rf /usr/share/HDP-templeton
   ```

   where

   - *$WEBHCAT_USER* is the WebHCat Service user. For example, hcat.
   - *$HDFS_USER* is the HDFS Service user. For example, hdfs.

4.17. Configure and Start Apache Pig

1. On all Apache Pig clients, run the following command:

   - **For RHEL/CentOS/Oracle Linux:**

     ```
     yum erase pig

     yum install pig
     ```

   - **For SLES:**

     ```
     zypper rm pig

     zypper install pig
     ```

2. Replace /etc/hive-webhcat/conf/ your configuration after upgrading.

   Copy /etc/pig/conf from the template to the conf directory in pig hosts.

4.18. Configure and Start Apache Sqoop

1. Replace your configuration after upgrading.

   Copy /etc/sqoop/conf from the template to the conf directory in sqoop hosts.

2. Upload the Apache Sqoop tarball to HDFS. As the <HDFS_User>, for example 'hdfs':

```
su -c "hdfs dfs -mkdir -p /hdp/apps/2.3.0.0-<$version>/sqoop"

hdfs su -c "hdfs dfs -chmod -R 555 /hdp/apps/2.3.0.0-<$version>/
sqoop"

hdfs su -c "hdfs dfs -chown -R hdfs:hadoop /hdp/apps/2.3.0.0-<
$version>/sqoop"

hdfs su -c "hdfs dfs -put /usr/hdp/2.3.0.0-<$version>/sqoop/
sqoop.tar.gz /hdp/apps/2.3.0.0-<$version>/sqoop/sqoop.tar.gz"

hdfs su -c "hdfs dfs -chmod 444 /hdp/apps/2.3.0.0-<$version>/
sqoop/sqoop.tar.gz" hdfs
```

4.19. Configure, Start, and Validate Apache Flume

Upgrade Apache Flume. On the Flume host machine, run the following command:

- For **RHEL/CentOS/Oracle Linux**:

  ```
  yum upgrade flume
  ```

- For **SLES**:

  ```
  zypper update flume

  zypper remove flume

  zypper se -s flume
  ```

  You should see Flume in the output.

  Install Flume:

  ```
  zypper install flume
  ```

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

By default on installation Flume does not start running immediately. To validate, replace your default conf/flume.conf with the provided flume.conf file, and restart Flume. See if the data is flowing by examining the destination.

Use this flume.conf file:

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

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

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

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

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

After starting Flume, check /tmp/flume to see if there are any files there. The files should contain simple sequential numbers.

After validating, stop Flume and revert changes to flume.conf to prevent your disk from filling up.

4.20. Configure, Start, and Validate Apache Mahout

Upgrade Apache Mahout. On the Mahout client machines, run the following command:

- For **RHEL/CentOS/Oracle Linux**:

  ```
  yum upgrade mahout
  ```

- For **SLES**:

  - `zypper remove mahout`
  - `zypper se -s mahout`

    You should see Mahout in the output.

  - Install Mahout:

    ```
    zypper up mahout
    ```

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

To validate mahout:

1. Create a test user:

   ```
   hadoop 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
   ```

4.21. Configure and Start Hue

For HDP 2.3, use the Hue version shipped with HDP 2.3. If you have a previous version of Hue, use the following steps to upgrade Hue.

1. Migrate the hue.ini setting from you rom 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_USER

   cd /var/lib/hue

   mv desktop.db desktop.db.old

   sqlite3 desktop.db < ~/hue_backup/desktop.bak

   exit
   ```

3. Synchronize the database:

   ```
   cd /usr/lib/hue

   source ./build/env/bin/activate

   hue syncdb

   deactivate
   ```

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

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

4.22. Finalize the Upgrade

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

Warning

You must verify your filesystem health before finalizing the upgrade. After you finalize an upgrade, you cannot roll back.

To finalize the upgrade process, run the following command as the $HDFS_USER:

```
hadoop dfsadmin -finalizeUpgrade
```

4.23. Install New HDP 2.3 Services

Install new HDP 2.3 Services:

- Atlas – a low-level service, similar to YARN, that provides metadata services to the HDP platform.
- SmartSense – a next generation subscription model that features upgrade and configuration recommendations.