Hortonworks Data Platform

Rolling Upgrade Guide

(October 13, 2015)

docs.cloudera.com

Hortonworks Data Platform: Rolling Upgrade Guide

The Hortonworks Data Platform, powered by Apache Hadoop, is a massively scalable and 100% open source platform for storing, processing and analyzing large volumes of data. It is designed to deal with data from many sources and formats in a very quick, easy and cost-effective manner. The Hortonworks Data Platform consists of the essential set of Apache 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. Overview

This document describes how to upgrade from HDP 2.2.0 to HDP 2.2.9 manually, using the HDP rolling upgrade feature. Rolling upgrade is supported on RHEL/CentOS/Oracle, SLES, Ubuntu, and Debian. If you have not yet upgraded to HDP 2.2, you will need to upgrade to HDP 2.2 before using the rolling upgrade feature.

If you use Ambari, see Automated HDP Stack Upgrade: HDP 2.2.0 to 2.2.x in the Ambari Documentation Suite.

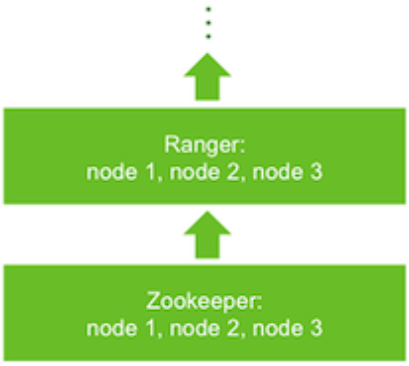
If you would like to add new components to your cluster, complete the upgrade process first, and then install the new components.

1.1. Introduction

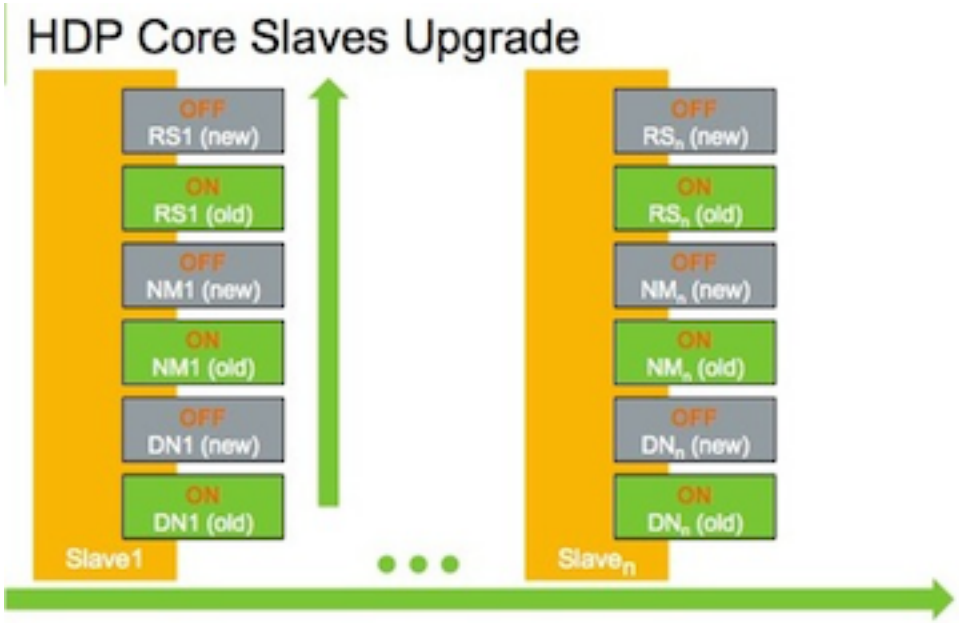
Rolling upgrade minimizes service disruption and degradation during the upgrade process. You can upgrade cluster components and restart upgraded services without taking the entire cluster down.

The rolling upgrade process follows a specific sequence based on component interactions and dependencies. Upgrade ZooKeeper first, followed by Ranger (if enabled on your cluster), core master processes, core slave processes, and onward through the stack, as shown in the following diagram.

You will usually upgrade each component across all of its deployed nodes before upgrading the next component in the sequence. For example, first you'll upgrade ZooKeeper across all nodes. Next you'll upgrade Ranger across all nodes:



The exception to this rule is the set of core slave processes. DataNodes, NodeManagers, and RegionServers must be updated on each node before proceeding to the next node:



1.2. Packaging

The rolling upgrade feature takes advantage of versioned packages.

Package names have the following format:

`<HDP-component>_<HDP-version>-<component-package>`

For example:

`hadoop_2_2_9_0_$BUILD-hdfs`

HDP 2.2 components are located at `/usr/hdp/<HDP-version>/<HDP-component>`. The `/usr/hdp/current` subdirectory contains links to the active HDP version for each component.

1.3. The upgrade process

First you will download the new software and locate it alongside the current software on each node. Next you will step through the component upgrade sequence, stopping each associated process, switching to the new software version, and restarting the process.

You'll use the `hdp-select` script to switch software versions. The script is included in the HDP repo, and requires root permissions. `hdp-select` sets up symlinks to `hdp-current` and modifies paths for configuration directories.

After each component upgrade you'll validate the results. When all components are running on the new software, check that your cluster is running to your satisfaction before finalizing the upgrade.

Although the rolling upgrade process minimizes service disruptions, the following components require service disruption during their portion of the rolling upgrade process:

- Falcon server
- Knox server
- Storm (Storm workloads will run during HDFS, YARN, HBase, and Hive upgrades)
- Storm on YARN via Slider
- Slider apps
- Hue server
- Accumulo

Where applicable, this is noted in the upgrade and downgrade instructions.

1.4. Reverting to prior state

If an upgrade step fails, there are two options for reverting to your prior state:

- *Rolling downgrade* restores software to the pre-upgrade version. The downgrade will preserve the user data that is generated during the upgrade process.

  The downgrade process starts with the component you most recently worked with during the upgrade process, and proceeds back down the stack. For example, if you upgraded ZooKeeper, Ranger, master and slave components, Hive, and Oozie, you would start the downgrade process with Oozie and then Hive. You would progress back through the stack, downgrading components in the reverse order, finishing with Ranger and then ZooKeeper.

- *Rollback* is an emergency procedure that restores your cluster services, components, metadata, and user data to a previously checkpointed state before rolling upgrade started. Rollback requires cluster downtime; it requires all services to be shut down before reverting to the previous version and restarting.

  The upgrade process includes several preparatory steps for rollback.

Downgrade and rollback operations are only available during the rolling upgrade process, before the upgrade is terminated (either by finalizing the upgrade, by downgrading, or by rolling back).

2. Preparation

Before starting the upgrade process, collect information about the topology of your cluster: the components, master processes, slave processes, and clients running on each node.

You will need root, administrative, or root-equivalent authorization to run `hdp-select` on all servers in the cluster, and superuser access to access and upgrade components.

On every node, make sure that the `/usr` directory can hold the new version of software in addition to your current version. Plan on 2.5 GB per version for an installation with all components.

The cluster will use your existing 2.2 locations for data and log files.

If you wish to add components to your cluster, complete the rolling upgrade process before adding the new components.

2.1. Cluster Prerequisites

To perform a manual rolling upgrade, your cluster must meet the following prerequisites:

Item	Description
Cluster Stack Version	Must be running the HDP 2.2 Stack
Cluster Target Version	All HDP nodes must have HDP 2.2.9 installed alongside 2.2.0. (See "Preparing the Cluster" for installation instructions.)
Services	All 2.2.0 services must be started and running. All previous upgrade operations must be finalized.
HDFS	NameNode HA must be enabled and running with an active namenode and standby namenode. No components should be in decommissioning or decommissioned state.
Hadoop, Hive, Oozie	Enable client retry for Hadoop, Hive, and Oozie.
YARN	Enable Work Preserving Restart (WPR). (Optional) Enable YARN ResourceManager High Availability
Shared client libraries	Shared client libraries must be loaded into HDFS.
Hive, Tez	Confirm configuration settings for rolling upgrade.

The following paragraphs describe component prerequisites in more detail. Examples assume that you are upgrading from 2.2.0 to 2.2.9.

- Enable HDFS NameNode High Availability. See NameNode High Availability for Hadoop (in the Hadoop High Availability Guide) for more information.
- Enable client retry properties for HDFS, Hive, and Oozie. These properties are not included by default, so you might need to add them to the site files.
  - For HDFS, set `dfs.client.retry.policy.enabled` to true in `hdfs-site.xml` on all nodes with HDFS services.
  - For Hive, specify `hive.metastore.failure.retries` and `hive.metastore.client.connect.retry.delay` in hive-site.xml (for example,

`/usr/hdp/2.2.0.0-2041/hive/conf/hive-site.xml`). The default value for retries is 24; the default for retry delay is 5s.

- For Oozie, `export OOZIE_CLIENT_OPTS="${OOZIE_CLIENT_OPTS} -Doozie.connection.retry.count=<number of retries>"` in `oozie-env.sh` (for example, `/usr/hdp/2.2.0.0-2041/oozie/conf/oozie-env.sh`). A typical value for number of retries is 5.

- Enable work-preserving ResourceManager/NodeManager restart in the `yarn-site.xml` file for each node. For more information, see Work-Preserving Restart in the YARN Resource Management Guide.

  Additional notes:

  - If `yarn.resourcemanager.work-preserving-recovery.scheduling-wait-ms` is set, the ResourceManager will wait for the specified number of milliseconds after each restart before accepting new jobs.

  - After editing ResourceManager settings in the `yarn-site.xml` file, restart the ResourceManager and all NodeManagers. (Changes will not take effect until you restart the processes.)

- (**Optional**) Enable YARN Resource Manager High Availability. Enabling RM HA will reduce the amount of service degradation while YARN is upgraded. If RM HA is not enabled, when the Resource Manager restarts your active jobs will pause and new job requests will wait to be scheduled. For more information, see Resource Manager High Availability for Hadoop.

- To prevent disruption to MapReduce, Tez, and Oozie jobs, your existing jobs must reference the client libraries of the version they started with. Make sure shared client Hadoop libraries are available from distributed cache. This was probably set up during the HDP 2.2.0 installation process. For more information, see Running Multiple MapReduce Versions Using the YARN Distributed Cache in the YARN Resource Management Guide.

- (**Optional**) When upgrading to HDP version 2.2.9 or later, remove the following two properties from the `hive-site.xml` configuration file (or set them to false):

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

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

- Make sure HiveServer2 is configured for rolling upgrade. Set or confirm the following server-side properties:

  - Set `hive.server2.support.dynamic.service.discovery` to `true`

  - Set `hive.zookeeper.quorum` to a comma-separated list of ZooKeeper host:port pairs in the ZooKeeper ensemble (e.g. `host1:port1, host2:port2, host3:port3`). By default this value is blank.

  - Add the `hive.zookeeper.session.timeout` property to the `hive-site.xml` file (if necessary), and specify the length of time that ZooKeeper will wait to hear from HiveServer2 before closing the client connection. The default value is 60 seconds.

- Set `hive.server2.zookeeper.namespace` to the value for the root namespace on ZooKeeper. (The root namespace is the parent node in ZooKeeper used by HiveServer2 when supporting dynamic service discovery.) Each HiveServer2 instance with dynamic service discovery enabled will create a `znode` within this namespace. The default value is `hiveserver2`.

  Note: you can specify the location of the `hive-site.xml` file via a HiveServer2 startup command line `--config` option:

  ```
  hive --config <my_config_path> --service hiveserver2
  ```

JDBC considerations:

- The JDBC driver connects to ZooKeeper and selects a HiveServer2 instance at random. When a JDBC client tries to pick up a HiveServer2 instance via ZooKeeper, the following JDBC connection string should be used:

  ```
  jdbc:hive2://
  <zookeeper_ensemble>/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=<hi
  ```

  where `<zookeeper_ensemble>` is a comma separated list of ZooKeeper host:port pairs, as described in the `hive.zookeeper.quorum` property.

  `<hiveserver2_zookeeper_namespace>` is the namespace on ZooKeeper under which HiveServer2 `znodes` are added. This instance is then used by the connecting client for her entire session.

- Check the following two settings to make sure that Tez is configured for rolling upgrade:

  - `tez.lib.uris` (in the `tez-site.xml` file) should contain only a single value pointing to a version-specific Tez tarball file. For 2.2 installations, the Tez app jars are in `/hdp/apps/${hp.version}`.

  - Set `tez.use.cluster.hadoop-libs` to `false`. (If true, the deployment will expect Hadoop jar files to be available on all nodes.

If Kerberos is enabled, it will continue to operate throughout the rolling upgrade process.

2.2. Preparing the Cluster

To prepare your cluster for rolling upgrade:

1. Download the new software to each node in the cluster.

2. Back up component metadata, so that the cluster can roll back to its earlier state if necessary.

3. Move new client libraries into HDFS.

During this time your cluster will continue to run on your current software version.

2.2.1. Download the new software onto each node in the cluster

The new version will reside alongside the existing version.

On each node of the cluster, download and set up the repo file for the new version. Confirm that the HDP repository is configured.

1. For RHEL, CentOS, or Oracle Linux, run the following commands.

   ```
   wget http://public-repo-1.hortonworks.com/HDP/<os>/2.x/
   updates/2.2.9.0/hdp.repo
   ```

   where <os> is `centos5` or `centos6`.

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

   ```
   yum repolist
   ```

2. For SLES:

   ```
   wget http://public-repo-1.hortonworks.com/HDP/<os>/2.x/
   updates/2.2.9.0/hdp.repo
   ```

   where <os> is `suse11sp3` or `sles11sp1`.

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

   ```
   zypper repos
   ```

3. For Ubuntu or Debian:

   ```
   wget http://public-repo-1.hortonworks.com/HDP/<os>/2.x/
   updates/2.2.9.0/hdp.list
   ```

   where <os> is `ubuntu12` or `debian6`.

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

   ```
   apt-get list
   ```

Install the new version of software on each node, based on the component topology of your cluster.

For example, some nodes will have core Hadoop components; others will have ZooKeeper processes. If only one node has the Hive Server deployed and running, install the new version of Hive on that node. Include the new version number.

1. For RHEL, CentOS, or Oracle Linux:

   ```
   yum install hive_2_2_9_0_$BUILD*
   ```

2. For SLES:

```
zypper install hive_2_2_9_0_$BUILD*
```

3. For Ubuntu and Debian:

```
apt-get install 'hive_2_2_9_0_$BUILD*'
```

Appendix B contains a partial list of HDP 2.2.9 package names.

At the end of this process your cluster will still be running your current software. The new software will be stored in a new directory on each node. You will see the following directories in `/usr/hdp`:

```
2.2.0.0-2041 2.2.9.0-$BUILD current
```

2.2.2. Back up component metadata

To ensure that your cluster can roll back to its earlier state if necessary, back up your component metadata before starting the rolling upgrade process.

Important

Back up your component metadata in the following order.

1. If Ranger is enabled, back up the Ranger policy database (see Upgrade Ranger in the Manual Upgrade Guide). If the audit records are in the Policy database, you can exclude the Audit table (`xa_access_audit`) from the export process.
2. Back up the Oozie Metastore database (see Getting Ready to Upgrade in the Manual Upgrade Guide).
3. Back up the Hive Metastore database (see Getting Ready to Upgrade in the Manual Upgrade Guide).
4. Create a backup snapshot of HBase in case of failure during the upgrade process:

   ```
   echo 'snapshot_all' | hbase shell
   ```

   Note the name of the HBase snapshot.
5. Back up Knox data. For example, if upgrading from 2.2.0.0-2041:

   ```
   su -l knox "mkdir -p /usr/hdp/2.2.0.0-2041/knox/data-bak"

   su -l knox "cp -r /usr/hdp/2.2.0.0-2041/knox/data /usr/
   hdp/2.2.0.0-2041/knox/data-bak"
   ```
6. Prepare HDFS for rolling upgrade:

   a. In case you decide to perform a rollback operation later, start the process to create an fsimage. Run the following command from $HDFS_USER (for example, `hdfs`):

      ```
      hdfs dfsadmin -rollingUpgrade prepare
      ```

      You'll see a message similar to the following:

```
PREPARE rolling upgrade ...
Preparing for upgrade. Data is being saved for rollback.
Run "dfsadmin -rollingUpgrade query" to check the status for proceeding
 with rolling upgrade
Block Pool ID: BP-288284058-240.0.0.10-1424468388939
Start Time: Mon Feb 23 14:29:01 PST 2015 (=1424730541827)
Finalize Time: <NOT FINALIZED>
```

b. Check the status of the rollback image preparation:

```
hdfs dfsadmin -rollingUpgrade query
```

c. View the query results. Continue to check status (step b) until you see the following message:

```
Proceed with rolling upgrade.
```

For example:

```
[hdfs@node-1 /]$ hdfs dfsadmin -rollingUpgrade query
QUERY rolling upgrade ...
Proceed with rolling upgrade:
Block Pool ID: BP-288284058-240.0.0.10-1424468388939
Start Time: Mon Feb 23 14:29:01 PST 2015 (=1424730541827)
Finalize Time: <NOT FINALIZED>
```

When you see the message to proceed, continue with the next step. On a small (two or three node) cluster this should not take more than a minute.

If the "proceed" message is delayed, check the NameNode logs to see if there are any errors or exceptions immediately after running the command. This will help you determine whether to wait, retry the process, or downgrade.

2.2.3. Move the new client libraries into HDFS

Several client libraries–such as MapReduce, Tez, Pig, Hive, and Sqoop libraries–are used by multiple components, shared via HDFS. Upload the new client libraries into HDFS. These libraries are versioned; they will not be used until they're enabled via the new clients for their respective components.

Note: you do not need to move the Oozie `sharelib` tarball into HDFS.

To move client tarballs into HDFS, run the following commands from any one of the cluster client nodes. Note that $HDFS_USER is defined as the HDFS superuser, such as `hdfs`.

```
su $HDFS_USER

# Load Tez client libraries into HDFS
hdfs dfs -mkdir -p /hdp/apps/2.2.9.0-$BUILD/tez/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/tez/lib/tez.tar.gz /hdp/apps/2.2.9.0-
$BUILD/tez/
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.2.9.0-$BUILD/tez
hdfs dfs -chmod -R 444 /hdp/apps/2.2.9.0-$BUILD/tez/tez.tar.gz

# Load MapReduce client libraries into HDFS
```

```
hdfs dfs -mkdir -p /hdp/apps/2.2.9.0-$BUILD/mapreduce/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/hadoop/mapreduce.tar.gz /hdp/apps/2.2.9.
0-$BUILD/mapreduce/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/hadoop-mapreduce/hadoop-streaming.jar /
hdp/apps/2.2.9.0-$BUILD/mapreduce/
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.2.9.0-$BUILD/mapreduce
hdfs dfs -chmod -R 444 /hdp/apps/2.2.9.0-$BUILD/mapreduce/mapreduce.tar.gz

# Load Hive client libraries into HDFS
hdfs dfs -mkdir -p /hdp/apps/2.2.9.0-$BUILD/hive/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/hive/hive.tar.gz /hdp/apps/2.2.9.0-
$BUILD/hive/hive.tar.gz
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.2.9.0-$BUILD/hive
hdfs dfs -chmod -R 444 /hdp/apps/2.2.9.0-$BUILD/hive/hive.tar.gz

# Load Pig client libraries into HDFS
hdfs dfs -mkdir -p /hdp/apps/2.2.9.0-$BUILD/pig/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/pig/pig.tar.gz /hdp/apps/2.2.9.0-$BUILD/
pig/
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.2.9.0-$BUILD/pig
hdfs dfs -chmod -R 444 /hdp/apps/2.2.9.0-$BUILD/pig/pig.tar.gz

# Load Sqoop client libraries into HDFS
hdfs dfs -mkdir -p /hdp/apps/2.2.9.0-$BUILD/sqoop/
hdfs dfs -put /usr/hdp/2.2.9.0-$BUILD/sqoop/sqoop.tar.gz /hdp/apps/2.2.9.0-
$BUILD/sqoop/
hdfs dfs -chown -R hdfs:hadoop /hdp
hdfs dfs -chmod -R 555 /hdp/apps/2.2.9.0-$BUILD/sqoop
hdfs dfs -chmod -R 444 /hdp/apps/2.2.9.0-$BUILD/sqoop/sqoop.tar.gz

# Log out
exit
```

Check that the files were copied:

```
su $HDFS_USER
hdfs dfs -ls /hdp/apps/2.2.9.0-$BUILD
exit
```

This should return results similar to the following:

Found 5 items

```
dr-xr-xr-x - hdfs hadoop 0 2015-03-03 17:05 /hdp/apps/2.2.9.0-$BUILD/hive
dr-xr-xr-x - hdfs hadoop 0 2015-03-03 17:04 /hdp/apps/2.2.9.0-$BUILD/mapreduce
dr-xr-xr-x - hdfs hadoop 0 2015-03-03 17:05 /hdp/apps/2.2.9.0-$BUILD/pig
dr-xr-xr-x - hdfs hadoop 0 2015-03-03 17:13 /hdp/apps/2.2.9.0-$BUILD/sqoop
dr-xr-xr-x - hdfs hadoop 0 2015-03-03 17:02 /hdp/apps/2.2.9.0-$BUILD/tez
```

3. Upgrading the Cluster

In this stage, services will be switched to the new software version.

Important

Perform all upgrade steps in the exact sequence listed in this section, for components deployed in your cluster.

1. Upgrade ZooKeeper
2. Upgrade Ranger (if Ranger is enabled on your cluster)
3. Upgrade core-cluster master processes, including HDFS JournalNodes and NameNode, YARN ResourceManager, and HBase Master(s)
4. Upgrade core-cluster slave processes: HDFS DataNodes, YARN NodeManagers, and HBase RegionServers
5. Upgrade non-core cluster components: Hive, Oozie, and Falcon
6. Upgrade Knox
7. Upgrade components outside of the cluster, such as Storm topologies that run on Storm standalone
8. Upgrade Slider
9. Upgrade clients on cluster (core and non-core) components, such as MapReduce, Tez, and Phoenix client
10. Finalize the rolling upgrade process. **Caution:** rollback and rolling downgrade operations are not possible after this step!

After upgrading each component, you'll validate that the component is running successfully.

Note

The `hdp-select` script requires root access. The instructions in this section assume that you start the upgrade process logged on as root.

3.1. Upgrade ZooKeeper

In this step you will upgrade and validate each ZooKeeper process, node by node.

1. Switch to the zookeeper_admin user (default = `zookeeper`) and run the following `stat` command for each node in the ZooKeeper quorum until you find the leader:

   ```
   su - zookeeper

   echo stat | nc <servername> <port> | grep Mode

   exit
   ```

   For example, to check nodes `hdp1` and `hdp2`, assuming they use default ZooKeeper server port 2181:

   ```
   [zookeeper@hdp1 ~] echo stat | nc hdp1 2181 | grep Mode
   Mode: follower
   [zookeeper@hdp1 ~] echo stat | nc hdp2 2181 | grep Mode
   Mode: leader
   [zookeeper@hdp1 ~] exit
   ```

   Note

   This is an optimization step. If the leader changes later, it won't impact the outcome of the upgrade process.

2. Upgrade ZooKeeper on each node. Start with follower nodes. Upgrade the leader node last.

   On each follower node, stop, switch, and start the daemon. Validate that the quorum has been reestablished before continuing with the next node:

a. Stop the ZooKeeper server, switch to the new version, and start the server:

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

hdp-select set zookeeper-server 2.2.9.0-$BUILD

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

b. Validate the upgrade: check that the ZooKeeper quorum has been reestablished before upgrading the next node. Create a new `znode` on the node just upgraded, and then list the created `znode`.

For example, start a `zkCli` process (ZooKeeper shell):

```
/usr/hdp/current/zookeeper-server/bin/zkCli.sh
```

Create a znode called `znode-1`. Associate the string "my_data" with the node:

```
[zk: localhost:2181(CONNECTED) 20] create /znode-1 my_data
Created /znode-1
```

List the `node-1` test directory:

```
[zk: localhost:2181(CONNECTED) 21] ls /znode-1
[]
```

Remove the test node and exit zkcli:

```
[zk: localhost:2181(CONNECTED) 22] delete /znode-1
[zk: localhost:2181(CONNECTED) 21] quit
Quitting...
[hdp@hdp1 /]#
```

c. Repeat the validation step until successful, or until you have completed ten attempts. If the test does not succeed after ten tries, see if the ZooKeeper process is running. If not, check ZooKeeper logs for an indication of the problem.

After upgrading ZooKeeper successfully on each follower node, perform the preceding upgrade steps for the leader node.

If the upgrade is unsuccessful or validations fail, follow the ZooKeeper downgrade steps in Downgrading the Cluster.

3.2. Upgrade Ranger

If Ranger is enabled in your cluster, follow the steps in this section before proceeding to step 3. Otherwise, skip this section. (Note: Ranger databases should be backed up before starting this step. See "Preparing the Cluster" for more information.)

3.2.1. Upgrade Ranger Admin

1. Shut down your current Ranger Admin process, switch to the new version, and start Ranger Admin:

   ```
   service ranger-admin stop

   hdp-select set ranger-admin 2.2.9.0-$BUILD

   service ranger-admin start
   ```

2. To validate the Ranger Admin upgrade, login to Ranger Admin via the web UI. The default user and password are `admin`, `admin`.

   ```
   http://${HOST}:6080
   ```

3.2.2. Upgrade Ranger UserSync

If enterprise user and group information needs to be synchronized with Ranger, the Apache Ranger UserSync component must be installed in RANGER_USERSYNC_HOST:

1. Shut down your current Ranger usersync process, switch to the new version, and start Ranger UserSync:

   ```
   service ranger-usersync stop

   hdp-select set ranger-usersync 2.2.9.0-$BUILD

   service ranger-usersync start
   ```

2. To validate the upgrade, go to /var/log/ranger/usersync and check for errors in the following files:

   ```
   usersync.log

   auth.log
   ```

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade Ranger."

3.3. Upgrade core-cluster master processes

In this step you will upgrade HDFS JournalNodes and NameNode, YARN-related master processes, and the HBase Master process.

3.3.1. Upgrade HDFS JournalNodes

Upgrade each JournalNode process one by one. The order does not matter.

Stop, switch, and start each JournalNode:

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

```
hdp-select set hadoop-hdfs-journalnode 2.2.9.0-$BUILD

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

3.3.2. Upgrade the HDFS NameNode HA Process Pair

When upgrading HDFS, you'll need to upgrade both NameNode processes:

1. Upgrade the standby process
2. Failover from active to standby
3. Upgrade the new standby process (formerly the active process)

In the following steps, "NN1" refers to your active NameNode process. "NN2" refers to your current standby NameNode process. (Note: the NameNode Web UI is at `http://namenode-name:50070/`.)

Switch the current standby process, NN2, to the new software version:

1. Shut down the current standby process NN2 and corresponding ZKFC process, switch NN2 to the new version, and start NN2. Note that the start command includes the "-rollingUpgrade started" option. NN2 will start and become the standby node again.

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

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

   hdp-select set hadoop-hdfs-namenode 2.2.9.0-$BUILD

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

2. Verify that NN2 is running successfully and in standby mode. Use the hdfs haadmin CLI to check state, or use the Web UI to check state and confirm the new version.

   If the service-ID of NN2 is configured as 'nn2', then the following command should return "standby":

   ```
   su - hdfs -c "hdfs haadmin -getServiceState nn2"
   ```

   (Service-IDs are defined in the dfs.ha.namenodes.<cluster-name> property in the hdfs-site.xml file. For more information, see Configure NameNode HA Cluster in the "NameNode High Availability" section of the Hadoop High Availability Guide.)

3. Start ZKFC:

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

4. Wait until NN2 is out of safe mode before proceeding. Review status info on the Web UI (`http://namenode-name:50070/`).

Failover from NN1 (active) to NN2. With NN1 now standby, switch NN1 to the new software version:

1. Force a failover from NN1 (currently active) to NN2, so that NN2 becomes active and NN1 becomes standby:

   ```
   su - hdfs -c "hdfs haadmin -failover <from-serviceid> <to-serviceid>"
   ```

   For example:

   ```
   su - hdfs -c "hdfs haadmin -failover nn1 nn2"
   Failover to NameNode at hdp1.lcl/172.16.226.128:8020 successful
   ```

2. Shut down NN1 and the corresponding ZKFC process, switch NN1 to the new version, and start NN1 as standby (again using the `-rollingUpgrade started` option):

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

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

   hdp-select set hadoop-hdfs-namenode 2.2.9.0-$BUILD

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

3. Verify that NN1 is running successfully and in standby mode. Use the `hdfs haadmin` CLI to check state. If, for example, the service-ID of NN1 is `nn1`, then the following command should return "standby":

   ```
   su - hdfs -c "hdfs haadmin -getServiceState nn1"
   ```

4. Start ZKFC:

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

3.3.3. Upgrade the MapReduce JobHistoryServer

This is the start of several steps related to the YARN rolling upgrade process. All running jobs should continue to make progress during and after the JobHistoryServer, YARN Timeline Service, ResourceMaster, and NodeManager upgrades, without task retries or failures due to the upgrade.

MapReduce applications will continue to use the same MapReduce version during the upgrade, even if their tasks are scheduled on nodes that are upgraded to the new version.

To upgrade the JobHistoryServer:

1. Shut down the JobHistoryServer process, switch to the new version, and start the JobHistoryServer process:

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

hdp-select set hadoop-mapreduce-historyserver 2.2.9.0-$BUILD

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

2. To validate the upgrade, check that the JobHistoryServer is running. (The default port is 19888.) Make sure BuildVersion (listed in **Application** -> **About**) matches your new version:

   ```
   http://<host>:<port>/jobhistory
   ```

3.3.4. Upgrade the YARN Timeline Service

Shut down the Timeline Service, switch to the newer version, and start the Timeline Service:

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

hdp-select set hadoop-yarn-timelineserver 2.2.9.0-$BUILD

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

3.3.5. Upgrade the YARN ResourceManager process or HA process pair

If High Availability is *not* enabled for the YARN ResourceManager, shut down the ResourceManager process, switch to the newer version, and start the ResourceManager process:

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

hdp-select set hadoop-yarn-resourcemanager 2.2.9.0-$BUILD

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

If High Availability *is* enabled for the YARN ResourceManager, you'll need to upgrade both ResourceManager processes, starting with the standby process. This is similar to the process you used for the NameNode pair:

1. Upgrade the standby process
2. Failover from active to standby
3. Upgrade the new standby process (formerly the active process)

In the following instructions, RM1 refers to the currently active ResourceManager. RM2 refers to the current standby ResourceManager.

1. Shut down the standby Resource Manager (RM2), switch to the newer version, and start RM2:

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

   hdp-select set hadoop-yarn-resourcemanager 2.2.9.0-$BUILD

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

   RM2 will start, and will become the standby ResourceManager.

2. To switch the standby ResourceManager, stop the active ResourceManager:

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

3. Shut down the new standby ResourceManager (RM1), switch to the new version, and start RM1:

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

   hdp-select set hadoop-yarn-resourcemanager 2.2.9.0-$BUILD

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

   RM1 will start and become the standby ResourceManager. For version info, see `http://<node-name>:8088/cluster/cluster`.

3.3.6. Upgrade HBase Master(s)

The last three steps for core-cluster master processes are for HBase.

Upgrade each HBase Master one by one. Shut down the HBase Master process, switch to the new version, and start the master process:

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop master"

hdp-select set hbase-master 2.2.9.0-$BUILD

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

During and after the upgrade process, all running jobs should continue to make progress without restarting.

3.3.7. Upgrade the HBase REST server

If you use the HBase REST server, complete the following steps. Otherwise, continue with the next step in the upgrade process.

The HBase REST server does not support rolling upgrade; you can't keep it running while you upgrade. To upgrade the REST server, stop and start the server:

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop rest"
```

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

(for example, port 8085)

When finished, verify that the REST server accepts traffic. 17000 is the default port for REST:

```
REST-gateway:17000/version
```

3.3.8. Upgrade the HBase Thrift server

If you use the HBase Thrift server, complete the following steps. Otherwise, continue with the next step in the upgrade process.

The HBase Thrift server does not support rolling upgrade; you cannot keep it running while you upgrade. To upgrade the Thrift server, stop and start the process.:

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop thrift"
```

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

To verify that the Thrift server accepts traffic, use a simple Thrift client to connect to the server.

Validate new core-masters with old core-slaves, old clients

At this point, validate the upgrade process by running the following tests. Make sure that these jobs execute successfully. If they succeed, proceed with rolling upgrade.

- MapReduce job
- Hive/Tez job
- `hadoop dfsadmin -report`
- `yarn –report`
- `hbase status`

See Appendix A for more information.

If the Ranger plugin is enabled for HBase, perform the following additional steps after upgrading HBase:

1. Login to Ranger via the Admin Web UI at `http://<RANGER_HOST>:6080`.
2. Go to **Audit**-> **Agent**. Make sure that the HBase agent has connected to Ranger Admin.

3. Go to **Audit** -> **Access**. Check the HBase log for errors.

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade core-cluster master processes."

3.4. Upgrade core slave nodes (HDFS, YARN, HBase)

Each core slave node in the cluster has an HDFS DataNode process, a YARN NodeManager process, and (optionally) an HBase RegionServer process.

Upgrade these processes as a set, one node at a time, before proceeding to the next node:

1. Upgrade HDFS DataNode, if present
2. Upgrade YARN NodeManager, if present
3. Upgrade HBase RegionServer, if present

3.4.1. On the core slave node, upgrade the HDFS DataNode

1. Shut down the current DataNode:

   ```
   su – hdfs -c "hdfs dfsadmin -shutdownDatanode
   <DATANODE_HOST:IPC_PORT> upgrade"
   ```

   For example:

   ```
   su - hdfs -c "hdfs dfsadmin -shutdownDatanode hdp1.lcl:8010 upgrade"
   Submitted a shutdown request to datanode hdp1.lcl:8010
   ```

2. Check to make sure the DataNode has stopped:

   ```
   su - hdfs -c "hdfs dfsadmin -getDatanodeInfo
   <DATANODE_HOST:IPC_PORT> upgrade"
   ```

   If the DataNode stopped successfully you'll see a series of retry messages, followed by "Datanode unreachable."

1. Switch to the new version and start the DataNode:

   ```
   hdp-select set hadoop-hdfs-datanode 2.2.9.0-$BUILD
   ```

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

2. To verify that the new DataNode joined the HDFS cluster, list live HDFS processes:

   ```
   su - hdfs -c "hdfs dfsadmin -report -live"
   ```

   The report should list the upgraded DataNode as a live node.

   Type **Control-C** to exit the utility.

3.4.2. On the core slave node, upgrade the YARN NodeManager

1. Shut down the current NodeManager process, switch to the new version, and start the new NodeManager process:

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

   hdp-select set hadoop-yarn-nodemanager 2.2.9.0-$BUILD

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

2. To verify that the NodeManager joined the cluster, check the status of the node:

   ```
   su - yarn -c "yarn node -list"
   ```

   Make sure that the node is listed as running. For example:

   ```
   [yarn@node-1 ~]$ yarn node -list
   15/02/24 18:17:20 INFO impl.TimelineClientImpl: Timeline service address:
   http://node-1.example.com:8188/ws/v1/timeline/
   15/02/24 18:17:20 INFO client.RMProxy: Connecting to ResourceManager at
   node-1.example.com/240.0.0.10:8032
   Total Nodes:1
   Node-Id                 Node-State Node-Http-Address          Number-of-
   Running-Containers
   node-1.example.com:45454 RUNNING    node-1.example.com:50060   0
   ```

3.4.3. On the core slave node, upgrade HBase RegionServer

1. Shut down the RegionServer, switch to the new version, and start the RegionServer:

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

   hdp-select set hbase-regionserver 2.2.9.0-$BUILD

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

2. Verify that the RegionServer joined the HBase cluster as a live server. The results of the following commands should list the upgraded RegionServer as a live server process.

   ```
   su – hbase

   echo 'status "detailed"' | hbase shell
   ```

3.4.4. Upgrade remaining nodes

Repeat steps 4.1, 4.2, and 4.3 for each remaining slave node in the cluster. After 20% of the core workers are upgraded, run the validation tests listed in the next subsection. If those pass, then it is safe to continue upgrading the rest of the worker nodes.

3.4.5. Validate new core-masters, core-workers with old clients

At the end of upgrading all worker nodes, then run the same set of validation tests to ensure that the cluster is functional.

At this point, validate the upgrade process by running the following tests. Make sure that these jobs execute successfully. If they succeed, proceed with rolling upgrade.

- MapReduce job
- Hive/Tez job
- `hadoop dfsadmin -report`
- `yarn –report`
- `hbase status`

See Appendix A for more information.

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade core slave nodes."

3.5. Upgrade non-core cluster components

These components are master-only, no slave/worker processes.

Upgrade non-core cluster components one at a time across all nodes, in the following order:

1. Upgrade Hive
2. Upgrade Oozie
3. Upgrade Falcon

3.5.1. Upgrade Hive

Hive consists of multiple services. Here is the general approach for the Hive rolling upgrade process:

- Upgrade each Hive Metastore server to the new version and restart it
- Start the new Hive Server 2 instance on each node and deregister the old Hive Server 2 instance
- Upgrade WebHCat to the new version and restart the WebHCat server
- Validate the upgrade process

3.5.1.1. Upgrade Hive Metastore(s)

Upgrade each Hive Metastore server one at a time across all Hive Metastore nodes. There is no order that needs to be followed. Before upgrading Hive, back up the database as described in "Prepare the Cluster."

Note: Existing connections to the Metastore will be dropped during this process, and there is a chance that running CLI jobs may fail. However, the Metastore will automatically retry any failed jobs.

1. Find the Hive Metastore server process ID and stop the process. For example:

```
# ps aux | grep hive
...
hive 129414 0.1 3.3 2002400 131332 ? . . . org.apache.hadoop.hive.metastore.
HiveMetaStore
...
# kill 129414
```

2. Switch to the new software version, and start the server:

```
hdp-select set hive-metastore 2.2.9.0-$BUILD

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

3.5.1.2. Upgrade Hive Server(s)

Upgrade each Hive Server process one node at a time across all Hive Server nodes. There is no order that needs to be followed.

Note that this process is slightly different than the stop–switch–start sequence for other components. You don't need to stop hive-server2; instead you will deregister the old copy after starting the new server process. The old version of hive-server2 will stop when it finishes its workload.

1. Switch hive-server2 to the new software version, start the new process, and deregister the old version of hive-server2. For example:

```
hdp-select set hive-server2 2.2.9.0-$BUILD

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

hive --service hiveserver2 --deregister 0.14.0.2.2.0.0-2041
```

2. To confirm deregistration, open a ZooKeeper CLI session and check files:

```
/usr/hdp/current/zookeeper-client/bin/zkCli.sh
[zk: localhost:2181(CONNECTING) 0] ls /
[hiveserver2, zookeeper]
[zk: localhost:2181(CONNECTED) 1] ls /hiveserver2
[serverUri=localhost:10000;version=0.14.0.2.2.9.0-$BUILD;sequence=
0000000000]
```

Before exiting the session (via `quit`) make sure that the old version is not listed.

3.5.1.3. Upgrade WebHCat (Templeton)

Upgrade the WebHCat server across all WebHCat nodes. Existing connections to WebHCat will be dropped, so there is a possibility that one or more running jobs may fail.

WebHCat will not automatically retry these failed jobs. The workaround is to retry the job manually from the client. Run the following commands from $WEBHCAT_USER (for example, hcat).

Shut down the WebHCat server, switch to the new version, and start the server:

```
sudo su -l $WEBHCAT_USER -c "/usr/hdp/current/hive-webhcat/sbin/
webhcat_server.sh stop"

hdp-select set hive-webhcat 2.2.9.0-$BUILD

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

Run the following validation tests; see Appendix A for more information:

- Hive/MapReduce job
- Hive/Tez job
- WebHCat smoke test

If the tests complete successfully, continue with the next step.

3.5.2. Upgrade Oozie

Upgrade each Oozie server one at a time.

1. On each Oozie server, shut down the server, switch to the new version, and start the server:

   ```
   /usr/hdp/current/oozie-server/bin/oozied.sh stop

   hdp-select set oozie-server 2.2.9.0-$BUILD

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

2. To validate the upgrade process, check Oozie service status.

   ```
   oozie admin -oozie http://<OOZIE_SERVER_HOST>:11000/oozie -
   status
   ```

   For example:

   ```
   # oozie admin -oozie http://node-1:11000/oozie -status
   System mode: NORMAL
   ```

3.5.3. Upgrade Falcon

1. Stop the Falcon server and (if installed) the Prism server:

   ```
   su - falcon -c "/usr/hdp/current/falcon-server/bin/falcon-stop"

   su - falcon -c "/usr/hdp/current/falcon-server/bin/prism-stop"
   ```

2. Switch to the new Falcon version and start the server:

   ```
   hdp-select set falcon-server 2.2.9.0-$BUILD

   hdp-select set falcon-client 2.2.9.0-$BUILD
   ```

3. Start the new Falcon server and (if installed) the Prism server:

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

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

4. Check the status of the new server. The following command should return status and port number:

   ```
   su - falcon ./bin/falcon-status
   ```

5. Run the Falcon smoke tests:

   ```
   su - falcon "/usr/lib/falcon/bin/falcon admin -version"

   Github >> ruCommon.py certification/HDTEsts/beaver/components/
   falcon/
   ```

See Appendix A for more information about smoke tests.

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade non-core cluster components." Otherwise, continue with the next upgrade step.

3.6. Upgrade Knox

There is service disruption when upgrading Knox. Upgrading Knox requires bringing Knox down completely, upgrading, and starting up again.

1. For each Knox server, stop the server, switch to the new version, and start the server:

   ```
   su -l knox "/usr/hdp/2.2.0.0-2041/knox/bin/gateway.sh stop"

   hdp-select set knox-server 2.2.9.0-$BUILD

   su -l knox "/usr/hdp/2.2.9.0-$BUILD/knox/bin/gateway.sh start"
   ```

2. Validate the installation:

   a. Check `/var/log/knox/gateway.log` for errors or successful start.

b. Validate cluster access through a WebHDFS API such as LISTSTATUS:

```
curl -ivk -u {user}:{password}

https://{knox-host}:8443/gateway/webhdfs/v1/tmp?op=LISTSTATUS
```

c. If the Knox admin service is deployed to the gateway instance, the version API should indicate the new version of the Knox code:

```
curl -ivk -u {adminuser}:{adminpassword}

https://{knox-host}:8443/gateway/admin/v1/version
```

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade Knox." Otherwise, continue with the next upgrade step.

3.7. Upgrade components outside of the cluster

The following components will be upgraded in this step:

- Storm
- Storm Cluster on YARN via Slider
- Kafka
- Hue

3.7.1. Upgrade Storm

Storm does not support rolling upgrade of the Storm service. It is best to stop the topology, update the client version to match the new version, and resubmit the topology:

1. Deactivate your currently running topology, using the CLI or Web UI:

   CLI: `storm deactivate <your-topology-name>`

   Web UI:

   a. To see a summary page for your topology, go to the Storm Web UI home page (`http://<storm-ui-server>:8080`) and click on your topology name.

   b. Look for a button to activate, deactivate, rebalance, or kill, on the top left side of the screen. Click "deactivate".

   c. In the popup window, enter the number of seconds after which the topology should be deactivated. Press OK.

2. Make sure that your topology has been deactivated:

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Deactivated".

3. Kill the deactivated topology, so that you can submit a new topology under the same name:

   CLI: `storm kill <your-topology-name>`

   Web UI:

   a. Go to the Storm Web UI home page and click on your topology name. This should open a summary page for your topology.

   b. Look for a button to activate, deactivate, rebalance, or kill, on the top left side of the screen. Click "kill".

   c. In the popup window, enter the number of seconds after which the topology should be killed. Press OK.

4. Make sure that your topology has been killed. Note: unlike the deactivate step, which shows the status of your topology, if your topology is killed you will not see an entry for your topology.

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Deactivated".

5. Update the component client's version to the new version.

6. Rebuild your topology jar with the new dependency. (For more information, see "Packaging Storm Topologies" in the Storm User's Guide.)

7. Upgrade the component to the new version.

8. Resubmit the topology using the new jar:

   ```
   storm jar <jar-path> <main class>
   ```

9. Confirm that your topology is active:

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Active".

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade components outside of the cluster." Otherwise, continue with the next upgrade step.

3.7.2. Upgrade Storm Cluster on YARN via Slider

Storm in Slider mode does not support rolling upgrade of the Storm service. It is best to stop the topology, update the client version to match the new version, and resubmit the topology.

1. Deactivate your currently running topology:

```
storm-slider --app <cluster-name> deactivate <your-topology-
name>
```

2. Make sure that your topology has been deactivated:

   ```
   storm-slider --app <cluster-name> list
   ```

3. Stop the deactivated topology, so that you can submit a new topology under the same name:

   ```
   storm-slider --app <cluster-name> kill <your-topology-name>
   ```

4. Make sure that your topology has been killed. Note: unlike the deactivate step, which shows the status of your topology, if your topology is killed you will not see an entry for your topology.

   ```
   storm-slider --app <cluster-name> list
   ```

5. Update the component client's version to the new version.

6. Rebuild your topology jar with the new dependency. (For more information, see "Packaging Storm Topologies" in the Storm User's Guide.)

7. Upgrade the component to the new version.

8. Resubmit the topology using the new jar:

   ```
   storm-slider --app <cluster-name> jar <jar-path> <main class>
   ```

9. Confirm that your topology is active:

   CLI: `storm-slider --app <cluster-name> list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Active".

If the upgrade process fails, follow the steps in "Downgrading the Cluster," starting with "Downgrade components outside of the cluster." Otherwise, continue with the next upgrade step.

3.7.3. Upgrade Kafka

Upgrade 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.

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 2.2.9.0-$BUILD

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

To verify that the Kafka daemon joined the cluster, run the associated validation steps in Appendix A.

If the upgrade process fails, follow the steps in "Downgrading the Cluster," starting with "Downgrade components outside of the cluster." Otherwise, continue with the next upgrade step.

3.7.4. Upgrade Hue

Hue does not support rolling upgrade, and it does not use versioned packaging. Hue should be upgraded to correspond with the appropriate version of the stack.

1. Shut down Hue:

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

2. If you are using the embedded SQLite database, perform a backup of the database before you upgrade Hue:

   ```
   su $HUE_USER

   mkdir ~/hue_backup

   cd /var/lib/hue

   sqlite3 desktop.db .dump > ~/hue_backup/desktop.bak

   exit
   ```

3. Upgrade the software. Hue uses old-style RPMs, not versioned RPMs:

   ```
   yum upgrade hue hue-common hue-server hue-beeswax hue-hcatalog
   hue-pig hue-oozie
   ```

4. If you are using the embedded SQLite database, restore your database after upgrading. To restore the database from a backup copy, 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
   /usr/lib/hue/build/env/bin/hue syncdb
   ```

5. Restart Hue. As root user, run the following command:

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

To validate the upgrade process, restart Hue and make sure that you can browse from the Hue UI (`http://hue.server:8000/`).

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade components outside of the cluster."

3.8. Upgrade Slider

During the upgrade process, existing Slider applications will continue to use the older version of the libraries.

To upgrade Slider so that new applications will use the newer version of Slider:

1. Switch to the new Slider software:

```
hdp-select set slider-client 2.2.9.0-$BUILD
```

2. Use the Slider version command to verify that Slider was upgraded properly:

```
./slider version
```

New application instances can be launched with the new version while older instances are running.

We recommend that you retain the older version of Slider, to ensure compatibility between client and Slider App Master for applications that are created and started using the older version.

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade Slider." Otherwise, continue with the upgrade process.

3.9. Upgrade clients on cluster (core and non-core) components

In this step, you will upgrade clients on gateway and cluster nodes. Components include Hadoop clients (HDFS client, YARN client, MapReduce, Tez, Hive, Pig), the HBase client, the Phoenix client, Flume, Mahout, and Sqoop.

These components do not supply services, so there is no need to follow the start-switch-stop process in earlier steps. You can simply switch the client software to the new version, in no specific order. Upgrade clients on all nodes where they exist, on edge/gateway nodes or on cluster nodes where they coexist with other services.

Here are sample commands:

```
hdp-select set hadoop-client 2.2.9.0-$BUILD

hdp-select set hbase-client 2.2.9.0-$BUILD

hdp-select set phoenix-client 2.2.9.0-$BUILD

hdp-select set mahout-client 2.2.9.0-$BUILD

hdp-select set sqoop-client 2.2.9.0-$BUILD
```

(Note: if the Sqoop metastore server is running, stop it before upgrading.)

The following subsections list additional information for MapReduce and Flume, followed by validation information.

3.9.1. Upgrade MapReduce Clients

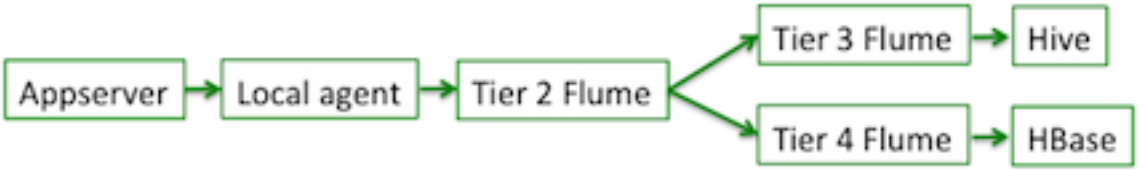
During the upgrade process, existing MapReduce jobs on the cluster should continue to run using the version with which they were launched.

Although you uploaded the new MapReduce tarballs to HDFS during the cluster preparation process, do not delete the older versions (and auxiliary jars) from HDFS until all jobs using them have completed.

3.9.2. Upgrade Flume

When performing a rolling upgrade of the entire Flume topology, we recommend that you start with the agents in the tier closest to the final data destination in the dataflow path. When you finish upgrading those agents, continue with the agents that are further away from the destination.

For example, consider the following Flume topology:



Here, tier 2 is writing to tier 3 and tier 4 Flume agents directly. Agents in tiers 3 and 4 should be upgraded first (in any order), followed by agents in tier 2, and finally the local agent.

To upgrade Flume complete the following steps, one Flume node at a time:

1. Stop all Flume agents on the node.
2. If the agents use File Channel or Spillable Channel, back up the data and checkpoint directories.
3. Switch to the new version:

   ```
   hdp-select set flume-server 2.2.9.0-$BUILD
   ```

4. Start the Flume agents, one by one.
5. To validate, make sure new data reaches the destination.
6. Optionally, uninstall the old Flume software once the new agent(s) have been running in a stable mode for a few days.

3.9.3. Validation

To validate the upgrade process for Hadoop, HBase, Phoenix, and Mahout clients, submit related smoke tests (see Appendix A).

To validate the Sqoop upgrade:

- Run the Sqoop smoke test.
- Invoke a Sqoop command and make sure that it responds correctly.
- The Sqoop `version` command should return the correct version.

If the upgrade process fails, follow the steps in Downgrading the Cluster, starting with "Downgrade clients on cluster (core and non-core) components." Otherwise, continue with the next upgrade step.

3.10. Finalize the rolling upgrade process

At this point in the process you have upgraded the binaries and metadata for all of your HDP components.

Run your application workloads on the cluster and make sure that your existing workloads function correctly. Do not proceed until you are confident that the cluster is operating to your satisfaction. Once you finalize the rolling upgrade process, you will not be able to revert to your previous version of HDP.

If there are validation issues at this point and you need to downgrade, follow the steps in Downgrading the Cluster, starting with "Downgrade components outside of the cluster."

HDFS backs up files before the upgrade. During the upgrade process, HDFS avoids deleting files (it creates hardlinks for deleted blocks for safety). Once you are satisfied with the upgrade and want to commit to it, run the following command to instruct HDFS to delete the blocks that were marked for deletion and also to perform deletes in the future.

Caution

This is irrevocable - you cannot revert back to your previous version of HDP after running this command.

```
hdfs dfsadmin -rollingUpgrade finalize
```

4. Downgrading the Cluster

If the validation process fails during any step in the upgrade process and the issue cannot be resolved, downgrade the stack.

Important: Start the downgrade process with the component you most recently upgraded or attempted to upgrade. Continue through the stack in reverse order compared with the upgrade process:

Here are the downgrade starting points based on your most recently upgraded component:

- Clients on cluster components (Flume, Sqoop, Pig, Tez, Phoenix, Mahout)
- Downgrade Slider [36]
- Components outside of the cluster (Hue, Kafka, Storm)
- Downgrade Knox [40]
- Non-core cluster components (Falcon, Oozie, Hive)
- Core slave nodes (DataNodes, NodeManager, RegionServer)
- Core master nodes (HDFS, YARN, HBase)
- Downgrade Ranger [52]
- Downgrade ZooKeeper [53]

4.1. Downgrade clients on cluster components

At this stage, the following components have been upgraded and need to be downgraded:

- Clients such as Flume, Sqoop, Pig, Tez, Phoenix, and Mahout
- Slider
- Components outside of the cluster: Hue, Kafka, Storm
- Knox

- Non-core cluster components: Falcon, Oozie, Hive
- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

Clients do not supply services, so you can simply switch the client software back to the earlier version, in no specific order. Downgrade clients on all nodes where they exist, on edge/gateway nodes or on cluster nodes where they coexist with other services.

Here are sample commands:

```
hdp-select set hadoop-client 2.2.0.0-2041

hdp-select set hbase-client 2.2.0.0-2041

hdp-select set phoenix-client 2.2.0.0-2041

hdp-select set mahout-client 2.2.0.0-2041
```

`hdp-select set sqoop-client 2.2.0.0-2041` (If the Sqoop metastore server is running, stop it before downgrading.)

The following subsections list additional instructions for Flume, followed by validation information.

4.1.1. Downgrade Flume

As with the upgrade process, when performing a rolling downgrade of Flume we recommend that you start with the agents in the tier closest to the final data destination in the dataflow path. When you finish downgrading those agents, downgrade agents that are further away from the destination.

1. Stop all Flume agents on the node.
2. If the agents use File channel or spillable channel, note that the checkpoint & data directories for the newer version of Flume might not be readable by the older version of Flume. If errors indicate that it does not work, you have one of two options:
   a. Restore the checkpoint and data directories from backup. If the newer version of Flume already processed some or all of the events, this could mean double delivery of those events–the downgraded flume will redeliver those events. This approach ensures, however, that no data is lost.
   b. Not recommended if it is important to not lose data: Start with empty checkpoint and data directories. Events that were backed up but not delivered by the new Flume agent, will be lost.
3. Switch back to the previous version of Flume on the node:

```
hdp-select set flume-server 2.2.2.0-2041
```

4. Start the Flume agents, one by one.

5. Validate by making sure new data is reaching the destination.

Validation

To validate the downgrade process for Hadoop, HBase, Phoenix, and Mahout clients, submit the smoke tests for MapReduce, HDFS, Phoenix, and Mahout (see Appendix A).

To validate Flume, make sure new data reaches the destination.

To validate Sqoop:

- Run the Sqoop smoke test (see Appendix A).
- Invoke a Sqoop command and make sure that it responds correctly.
- The Sqoop `version` command should return the correct version.

Continue with the next downgrade step.

4.2. Downgrade Slider

At this stage, the following components have been upgraded and need to be downgraded:

- Slider
- Components outside of the cluster: Hue, Kafka, Storm
- Knox
- Non-core cluster components: Falcon, Oozie, Hive
- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

To downgrade Slider, just uninstall the version to be removed. Use the Slider version command to verify that Slider was downgraded:

```
./slider version
```

During the downgrade process, existing Slider applications will continue to run, using the version of the libraries they were launched with. We recommend that you retain the newer version of Slider until applications finish that were started with the newer version.

Continue with the next downgrade step.

4.3. Downgrade components outside of the cluster

At this stage, the following components have been upgraded and need to be downgraded:

- Components outside of the cluster: Hue, Kafka, Storm
- Knox
- Non-core cluster components: Falcon, Oozie, Hive
- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

4.3.1. Downgrade Hue

To downgrade hue, shut down Hue, downgrade the software, and restart Hue.

Caution

Hue does not support rolling downgrade. The SQLite database will be reset during the downgrade process. This can cause loss of Hue user data.

1. Shut down Hue:

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

2. Downgrade the software. Hue uses old-style RPMs, not versioned RPMs:

   ```
   yum downgrade hue hue-common hue-server hue-beeswax hue-hcatalog
   hue-pig hue-oozie
   ```

3. Restart Hue. As root user, run the following command:

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

Validation

To validate the downgrade process, restart Hue and make sure that you can browse from the Hue UI:

```
http://hue.server:8000/
```

Continue with the next step.

4.3.2. Downgrade 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.

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"
```

To verify that the Kafka daemon rejoined the cluster, run the validation steps in Appendix A.

Continue with the next step.

4.3.3. Downgrade Storm Cluster on YARN via Slider

Storm in Slider mode does not support rolling downgrade of the Storm service. It is best to stop the topology, update the client version to match the previous version, and resubmit the topology.

1. Deactivate your currently running topology:

   ```
   storm-slider --app <cluster-name> deactivate <your-topology-name>
   ```

2. Make sure that your topology has been deactivated:

   ```
   storm-slider --app <cluster-name> list
   ```

3. Kill the deactivated topology, so that you can submit a new topology under the same name:

   ```
   storm-slider --app <cluster-name> kill <your-topology-name>
   ```

4. Make sure that your topology has been killed. Note: unlike the deactivate step, which shows the status of your topology, if your topology is killed you will not see an entry for your topology.

   ```
   storm-slider --app <cluster-name> list
   ```

5. Switch the component client's version to the previous version.

6. Rebuild your topology jar with the new dependency. (For more information, see "Packaging Storm Topologies" in the Storm User's Guide.)

7. Upgrade the component to the previous version.

8. Resubmit the topology using the previous jar:

```
storm-slider --app <cluster-name> jar <jar-path> <main class>
```

9. Confirm that your topology is active:

```
storm-slider --app <cluster-name list
```

If the downgrade process fails, follow the steps in "Downgrade your Cluster," starting with "Downgrade components outside of the cluster." Otherwise, continue with the next downgrade step.

4.3.4. Downgrade Storm

Storm does not support rolling downgrade of the Storm service. It is best to stop the topology, update the client version to match the previous version, and resubmit the topology.

1. Deactivate your currently running topology, using the CLI or Web UI:

   CLI: `storm deactivate <your-topology-name>`

   Web UI:

   a. To see a summary page for your topology, go to the Storm Web UI home page (`http://<storm-ui-server>:8080`) and click on your topology name.

   b. Look for a button to activate, deactivate, rebalance, or kill, on the top left side of the screen. Click "deactivate".

   c. In the popup window, enter the number of seconds after which the topology should be deactivated. Press OK.

2. Make sure that your topology has been deactivated:

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Deactivated".

3. Kill the deactivated topology, so that you can submit a new topology under the same name:

   CLI: `storm kill <your-topology-name>`

   Web UI:

   a. Go to the Storm Web UI home page and click on your topology name. This should open a summary page for your topology.

   b. Look for a button to activate, deactivate, rebalance, or kill, on the top left side of the screen. Click "kill".

   c. In the popup window, enter the number of seconds after which the topology should be killed. Press OK.

4. Make sure that your topology has been killed. Note: unlike the deactivate step, which shows the status of your topology, if your topology is killed you will not see an entry for your topology.

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Deactivated".

5. Switch the component client's version to the previous version.

6. Rebuild your topology jar with the new dependency. (For more information, see "Packaging Storm Topologies" in the Storm User's Guide.)

7. Upgrade the component to the previous version.

8. Resubmit the topology using the previous jar:

   ```
   storm jar <jar-path> <main class>
   ```

9. Confirm that your topology is active:

   CLI: `storm list`

   Web UI: Go to the Storm Web UI home page. You should see topology status in the list of topologies. Make sure the status column lists "Active".

If the downgrade process fails, follow the steps in "Downgrade your Cluster," starting with "Downgrade components outside of the cluster." Otherwise, continue with the next downgrade step.

4.4. Downgrade Knox

At this stage, the following components have been upgraded and need to be downgraded:

- Knox
- Non-core cluster components: Falcon, Oozie, Hive
- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

There is service disruption when downgrading Knox. Downgrading Knox requires bringing Knox down completely, downgrading, and starting up again.

1. Shut down all Knox servers:

   ```
   su -l knox "/usr/hdp/2.2.9.0-$BUILD/knox/bin/gateway.sh stop"
   ```

2. Revert any changes to the data that were made after the upgrade process started, or restore the earlier version of your data from backup.

3. Switch to the previous Knox software version, and then start the server:

```
hdp-select set knox-server 2.2.0.0-2041
```

```
su -l knox "/usr/hdp/2.2.0.0-2041/knox/bin/gateway.sh start"
```

4. Validate the installation:

   a. Check `/var/log/knox/gateway.log` for errors or successful start.

   b. Validate cluster access through a WebHDFS API such as `LISTSTATUS`:

```
curl -ivk -u {user}:{password}
```

```
https://{knox-host}:8443/gateway/webhdfs/v1/tmp?op=LISTSTATUS
```

   c. If the Knox admin service is deployed to the gateway instance, the version API should indicate the previous version of the Knox binaries:

```
curl -ivk -u {adminuser}:{adminpassword}
```

```
https://{knox-host}:8443/gateway/admin/v1/version
```

Continue with the next downgrade step.

4.5. Downgrade non-core cluster components

At this stage, the following components have been upgraded and need to be downgraded:

- Non-core cluster components: Falcon, Oozie, Hive
- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

4.5.1. Downgrade Falcon

1. Stop the Falcon server and, if installed, the Prism server:

```
su - falcon -c "/usr/hdp/current/falcon-server/bin/falcon-stop"
```

```
su - falcon -c "/usr/hdp/current/falcon-server/bin/prism-stop"
```

2. Switch to the previous Falcon version:

```
hdp-select set falcon-server 2.2.0.0-2041
```

3. Start the Falcon server and, if installed, the Prism server:

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

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

4. Check the status of the server. The following command should return status and port number:

   ```
   su - falcon ./bin/falcon-status
   ```

5. Run the falcon smoke tests listed in Appendix A, and perform additional validation if necessary.

Continue with the next downgrade step.

4.5.2. Downgrade Oozie

Downgrade each Oozie server one at a time.

1. Shut down the Oozie server, switch to the previous version, and start the server:

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

   hdp-select set oozie-server 2.2.0.0-2041

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

2. To validate the downgrade process, check Oozie service status:

   ```
   oozie admin -oozie http://<OOZIE_SERVER_HOST>:11000/oozie -
   status
   ```

   For example:

   ```
   # oozie admin -oozie http://node-1:11000/oozie -status
   System mode: NORMAL
   ```

   In addition, run Oozie smoke tests (see Appendix A).

Continue with the next downgrade step.

4.5.3. Downgrade Hive

The Hive downgrade process reverses the upgrade process:

- Downgrade WebHCat
- Start the previous Hive Server 2 instance
- Deregister the new Hive Server 2 instance
- Downgrade Hive Metastore servers

When finished, validate the downgrade process.

4.5.3.1. Downgrade WebHcat (Templeton)

Downgrade the WebHCat server as described below. Existing connections to WebHCat will be dropped, so there is a possibility that one or more running jobs may fail.

WebHCat will not automatically retry these failed jobs. The workaround is to retry the job manually from the client.

Run the following commands from $WEBHCAT_USER (for example, `hcat`).

1. Shut down the WebHCat server, switch to the previous version, and start the server:

```
sudo su -l $WEBHCAT_USER -c "/usr/hdp/current/hive-webhcat/sbin/
webhcat_server.sh stop"

hdp-select set hive-webhcat 2.2.0.0-2041

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

2. Run the following validation tests:
   - Hive/MapReduce job
   - Hive/Tez job
   - WebHCat smoke test

See Appendix A for smoke test information.

4.5.3.2. Downgrade Hive Server(s)

Downgrade each Hive Server process one node at a time, as described below. There is no order that needs to be followed.

Note

You don't need to stop `hive-server2`; instead you will deregister the newer version after starting the previous version of the server. The newer version of `hive-server2` will stop when it finishes its workload.

1. Switch `hive-server2` to the previous software version, start the previous version, and deregister the new version. For example:

```
hdp-select set hive-server2 2.2.0.0-2041

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

hive --service hiveserver2 --deregister 0.14.0.2.2.9.0-$BUILD
```

2. To confirm deregistration, open a ZooKeeper CLI session and check files:

```
/usr/hdp/current/zookeeper-client/bin/zkCli.sh
```

```
[zk: localhost:2181(CONNECTING) 0] ls /
[hiveserver2, zookeeper]
[zk: localhost:2181(CONNECTED) 1] ls /hiveserver2
[serverUri=localhost:10000;version=0.14.0.2.2.0.0-2041;sequence=0000000000]
```

Before exiting the session (using `quit`) make sure that the newer version is no longer listed.

4.5.3.3. Downgrade Hive Metastore(s)

Downgrade each Hive Metastore server one at a time, as described below. There is no order that needs to be followed.

Note: Existing connections to the Metastore will be dropped during this process, and there is a chance that running CLI jobs may fail. However, the Metastore will automatically retry any failed jobs.

1. Find the Hive Metastore server process ID and stop the process. For example:

```
# ps aux | grep hive
...
hive 129414 0.1 3.3 2002400 131332 ...
org.apache.hadoop.hive.metastore.HiveMetaStore
...
# kill 129414
```

2. Switch to the previous software version and start the server:

```
hdp-select set hive-metastore 2.2.0.0-2041

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

3. Validate the downgrade process: make sure that the following jobs run successfully.
   - Hive/MapReduce job
   - Hive/Tez job

Continue with the next downgrade step.

4.6. Downgrade core slave nodes

At this stage, the following components have been upgraded and need to be downgraded:

- Core slave nodes: DataNodes, NodeManager, RegionServer
- Core masters: HDFS, YARN, HBase
- Ranger

- ZooKeeper

Downgrade these processes as a set, one node at a time, before proceeding to the next node:

1. Downgrade HDFS DataNode, if present
2. Downgrade YARN NodeManager, if present
3. Downgrade HBase RegionServer, if present

4.6.1. Downgrade the HBase RegionServer

1. Shut down the RegionServer, switch to the previous version, and start the server:

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

hdp-select set hbase-regionserver 2.2.0.0-2041

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

2. Verify that the RegionServer joined the HBase cluster as a live server. The results of the following command sequence should list the downgraded RegionServer as a live server process:

```
su – hbase

echo 'status "detailed"' | hbase shell
```

Continue with the next downgrade step.

4.6.2. Downgrade the YARN NodeManager

Stop the NodeManager process, switch versions, and restart the NodeManager:

1. Shut down the current NodeManager process, switch to the previous version, and start the NodeManager process:

```
su – yarn "/usr/hdp/current/hadoop-yarn-nodemanager/sbin/yarn-
daemon.sh stop nodemanager"

hdp-select set hadoop-yarn-nodemanager 2.2.0.0-2041

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

2. To verify that the NodeManager joined the cluster, check the status of the node:

```
yarn node –list
```

Make sure the node is listed as running. For example:

```
[yarn@node-1 ~]$ yarn node -list
15/02/24 18:17:20 INFO impl.TimelineClientImpl: Timeline service address:
 http://node-1.example.com:8188/ws/v1/timeline/
15/02/24 18:17:20 INFO client.RMProxy: Connecting to ResourceManager at
 node-1.example.com/240.0.0.10:8032
Total Nodes:1
Node-Id                 Node-State Node-Http-Address         Number-of-
Running-Containers
node-1.example.com:45454 RUNNING    node-1.example.com:50060  0
```

Continue with the next downgrade step.

4.6.3. Downgrade HDFS DataNodes

1. Shut down the current DataNode:

   ```
   su – hdfs -c "hdfs dfsadmin -shutdownDatanode
   <DATANODE_HOST:IPC_PORT> upgrade"
   ```

   For example:

   ```
   [hdp@node-1 ~]# su - hdfs -c 'hdfs dfsadmin -shutdownDatanode node-1.
   example.com:8010 upgrade'
   Submitted a shutdown request to datanode node-1.example.com:8010
   ```

2. Check to make sure the DataNode has stopped:

   ```
   su – hdfs -c "hdfs dfsadmin -getDatanodeInfo
   <DATANODE_HOST:IPC_PORT>"
   ```

   If the DataNode stopped successfully you'll see a series of retry messages, followed by `Datanode unreachable`.

3. Switch to the previous DataNode version, and then start the DataNode:

   ```
   hdp-select set hadoop-hdfs-datanode 2.2.0.0-2041
   ```

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

4. To verify that the DataNode joined the HDFS cluster, list live HDFS processes:

   ```
   su - hdfs -c "hdfs dfsadmin -report -live"
   ```

   This should list the downgraded DataNode as a live node.

   Type Control-C to exit the utility.

4.6.4. Validate the downgrade process

At this point, validate the downgrade process by running the following tests. Make sure that these jobs execute successfully.

- MapReduce job

- Hive/Tez job
- `hadoop dfsadmin -report`
- `yarn –report`
- `hbase status`

See Appendix A for specific commands.

Continue with the next downgrade step.

4.7. Downgrade core-cluster master processes

At this stage, the following components have been upgraded and need to be downgraded:

- Core masters: HDFS, YARN, HBase
- Ranger
- ZooKeeper

4.7.1. Downgrade the HBase Thrift server

If you use the HBase Thrift server, complete the following steps. Otherwise, continue with the next step in the upgrade process.

The HBase Thrift server does not support rolling upgrade; you cannot keep it running while you downgrade. To downgrade the Thrift server, stop and start the process.

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop thrift"
```

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

To verify that the Thrift server accepts traffic, use a simple Thrift client to connect to the server.

Continue with the next downgrade step.

4.7.2. Downgrade HBase REST Server

If you use the HBase REST server, follow these steps. Otherwise, continue with the next step in the upgrade process.

The HBase REST server does not support rolling downgrade; you can't keep it running as you downgrade. To downgrade the REST server, stop and start the process:

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop rest"
```

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

(for example, port 8085)

When finished, verify that the REST server accepts traffic. 17000 is the default port for REST:

```
REST-gateway:17000/version
```

Continue with the next downgrade step.

4.7.3. Downgrade HBase Master(s)

During the downgrade process, running jobs should continue to make progress without restarting.

If there is an unforeseen error during upgrade, run the following command to restore data:

```
echo "snapshot_restore '20140917'" | hbase shell
```

Downgrade each HBase Master one by one. Shut down the HBase Master process, switch to the previous version, and start the process:

```
su - hbase -c "/usr/hdp/current/hbase-master/bin/hbase-daemon.sh
stop master"

hdp-select set hbase-master 2.2.0.0-2041

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

Continue with the next downgrade step.

4.7.4. Downgrade the YARN ResourceManager process or HA process pair

This is the start of several steps related to the YARN downgrade process. All running jobs should continue to make progress during and after you downgrade JobHistoryServer, YARN Timeline Service, ResourceMaster, and NodeManager processes, without task retries or failures due to the downgrade.

MapReduce applications will continue to use the same MapReduce version during the downgrade.

If High Availability is *not* enabled for the YARN ResourceManager, downgrade the single ResourceManager process as follows:

Shut down the ResourceManager process, switch to the previous version, and start the process:

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

hdp-select set hadoop-yarn-resourcemanager 2.2.0.0-2041
```

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

If High Availability *is* enabled for the YARN ResourceManager you'll need to downgrade both ResourceManager processes, starting with the standby process.

In the following instructions, RM1 refers to the currently active ResourceManager. RM2 refers to the current standby ResourceManager.

1. Stop, downgrade, and start the standby Resource Manager (RM2):

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

   hdp-select set hadoop-yarn-resourcemanager 2.2.0.0-2041

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

   RM2 will start, and will become the standby Resource Manager again.

2. To switch the standby ResourceManager to be the active ResourceManager, stop the active ResourceManager:

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

3. Shut down RM1, switch to the previous version, and start RM1 (as standby):

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

   hdp-select set hadoop-yarn-resourcemanager 2.2.0.0-2041

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

   RM1 will start and become the standby ResourceManager. For version info, see `http://<node-name>:8088/cluster/cluster`.

Continue with the next downgrade step.

4.7.5. Downgrade the YARN Timeline Service

Shut down the Timeline Service, switch to the previous version, and start the service:

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

hdp-select set hadoop-yarn-timelineserver 2.2.0.0-2041

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

Continue with the next downgrade step.

4.7.6. Downgrade the MapReduce JobHistoryServer

Note

Retain the newer versions of MapReduce tarballs and auxiliary jars on HDFS until all jobs using them have completed on the cluster.

Shut down the JobHistoryServer process, switch to the previous version, and start the process:

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

hdp-select set hadoop-mapreduce-historyserver 2.2.0.0-2041

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

To validate the downgrade, check that the JobHistoryServer is running. The default port is 19888. Make sure BuildVersion (listed in Application -> About) matches the previous version:

```
http://<host>:<port>/jobhistory
```

Continue with the next downgrade step.

4.7.7. Downgrade the HDFS NameNode HA Process Pair

During the downgrade process, you will start the NameNode process with the regular start command (without the -rollingUpgrade option used during the upgrade process).

In the following steps, "NN1" refers to your active NameNode process, and "NN2" refers to your current standby NameNode process. (The NameNode Web UI is at `http://namenode-name:50070/`.)

Switch the current standby process, NN2, to the previous software version:

1. Shut down NN2 and the corresponding ZKFC process, switch NN2 to the previous version, and start NN2.

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

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

   hdp-select set hadoop-hdfs-namenode 2.2.0.0-2041

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

2. Verify that NN2 is running successfully and in standby mode. Use the `hdfs haadmin` CLI to check state. If the service-ID of NN2 is `nn2`, then the following command should return "standby":

```
> hdfs haadmin -getServiceState nn2
```

3. Start ZKFC:

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

4. Wait until NN2 is out of safe mode before proceeding. To review status, see the Web GUI at `http://${HOST}:50070`.

Next, failover from NN1 to NN2, and downgrade NN1:

1. Force a failover from NN1 (currently active) to NN2, so that NN1 becomes the standby process:

```
>hdfs haadmin -failover --forceactive <from-serviceid> <to-
serviceid>
```

2. Shut down NN1 and the corresponding ZKFC process, switch to the previous version, and start NN1:

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

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

hdp-select set hadoop-hdfs-namenode 2.2.0.0-2041

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

3. Verify that NN1 is running successfully and in standby mode. Use the `hdfs haadmin` CLI to check state. If the service-ID of NN1 is configured as `nn1`, then the following command should return "standby":

```
> hdfs haadmin -getServiceState nn1
```

4. Start ZKFC:

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

Continue with the next downgrade step.

4.7.8. Downgrade HDFS JournalNode(s)

Downgrade each JournalNode process, one by one. The order does not matter.

Stop the JournalNode process, switch to the previous version, and start the process:

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

```
hdp-select set hadoop-hdfs-journalnode 2.2.0.0-2041
```

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

See Appendix A for HDFS validation tasks.

Continue with the next downgrade step.

4.7.9. Validation

At this point, validate the upgrade process by running the following tests:

- MapReduce job
- Hive/Tez job
- `hadoop dfsadmin -report`
- `yarn –report`
- `hbase status`

See Appendix A for specific commands.

4.8. Downgrade Ranger

If Ranger is enabled in your cluster, follow the steps in this section to downgrade it. Otherwise, skip this section and proceed to the next section.

At this stage, the following components have been upgraded and need to be downgraded:

- Ranger
- ZooKeeper

4.8.1. Downgrade Ranger UserSync

Shut down the Ranger User Sync process, switch to the previous version, and start Ranger User Sync:

```
service ranger-usersync stop
```

```
hdp-select set ranger-usersync 2.2.0.0-2041
```

```
service ranger-usersync start
```

To validate the User Sync downgrade, go to `/var/log/ranger/usersync` and check for errors in the following files:

- `usersync.log`
- `auth.log`

Continue with the next downgrade step.

4.8.2. Downgrade Ranger Admin

Shut down the Ranger Admin process, switch to your previous version, and start Ranger Admin:

```
service ranger-admin stop

hdp-select set ranger-admin 2.2.0.0-2041

service ranger-admin start
```

To validate the Ranger Admin upgrade, login to Ranger Admin via the web UI, `http://${HOST}:6080`. The default user and password are `admin`, `admin`.

Go to `/var/log/ranger/admin` and check error logs:

- `catalina.out`
- `xa_portal.log`

4.8.3. Validate Ranger HBase Plugin

If the Ranger plugin is enabled for HBase, perform the following steps:

1. Login on Ranger Admin Web Portal:

   ```
   http://<RANGER_HOST>:<PORT>
   ```

   (The default port is 6080.)

2. Go to Audit- > Agent. Make sure that the HBase agent has connected to Ranger Admin.

3. Go to Audit -> Access Log. Check the HBase logs for errors.

After validating the Ranger downgrade, continue with the next downgrade step.

4.9. Downgrade ZooKeeper

ZooKeeper is the final component in the downgrade process.

In this step you will downgrade and validate each ZooKeeper process, node by node.

1. Sign on as the zookeeper_admin user (default = `zookeeper`) and run the following `stat` command for each node in the ZooKeeper quorum until you find the leader:

   ```
   su - zookeeper

   echo stat | nc <servername> <port> | grep Mode

   exit
   ```

   For example, to check nodes `hdp1` and `hdp2`, assuming they use default ZooKeeper server port 2181:

```
[zookeeper@hdp1 ~] echo stat | nc hdp1 2181 | grep Mode
Mode: follower
[zookeeper@hdp1 ~] echo stat | nc hdp2 2181 | grep Mode
Mode: leader
[zookeeper@hdp1 ~] exit
```

As with the related upgrade step, this is an optimization step. If the leader changes later, it won't impact the outcome of the downgrade process.

2. Downgrade and restart each node, as follows. Downgrade the leader last.

   a. Stop the ZooKeeper server, switch to the previous version, and start the server:

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

   hdp-select set zookeeper-server 2.2.0.0-2041

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

   b. Make sure that the ZooKeeper quorum has been reestablished before downgrading the next node. To check this, create a new znode on the node just downgraded, and then list the created znode.

   For example, start a `zkcli` process (ZooKeeper shell) that points to the local host (`zkcli` is in /usr/hdp/current/zookeeper-server/bin):

   ```
   ./zkCli.sh localhost
   ```

   Create a znode called `znode-1`. Associate the string `my_data` with the node:

   ```
   [zk: node-1:2181(CONNECTED) 20] create /znode-1 my_data
   Created /znode-1
   ```

   List the node-1 test directory:

   ```
   [zk: node-1:2181(CONNECTED) 21] ls /znode-1
   []
   ```

   Remove the test node and exit `zkcli`:

   ```
   [zk: node-1:2181(CONNECTED) 22] delete /znode-1
   [zk: node-1:2181(CONNECTED) 21] quit
   [hdp@node-1 bin]#
   ```

   c. Repeat the validation step until successful or until you have completed ten attempts. If the test does not succeed after ten tries, see if the ZooKeeper process is running. If not, check ZooKeeper logs for an indication of the problem.

3. After downgrading ZooKeeper successfully on each node, restart the leader node.

5. Initiating Rollback

Rollback is the ability to revert to the original state of the HDP cluster before a rolling upgrade was started.

A rollback operation reverts your cluster to the original binaries, metadata, and user data. Rollback removes any new metadata and user data that is generated after the upgrade process begins.

The rollback operation requires several prerequisite steps, listed earlier in the Rolling Upgrade Guide. These steps include:

- Backing up component metadata for HDFS, HBase, Hive, Oozie, and Ranger
- Using the `-rollingUpgrade` option when preparing a rollback image for HDFS, and when restarting upgraded HDFS NameNodes

Rollback requires all services in the cluster to be shut down and restarted. It is not a rolling process.

For assistance with a rollback operation, contact Hortonworks support.

6. Appendix A

This section contains information and links for validating HDP components.

The following links are in "Installing HDP Manually":

Table 6.1. Validation Links

Component	Tests
Falcon	Submit a entity, dataset and process to Falcon. Validate that the submission succeeded.
HBase	Check HBase status from the HBase shell.
HDFS	Write data to HDFS, list contents.
Hive	List databases, create new table. Run sample commands through the Hive CLI shell and Hive Server 2.
Hue	Access current configuration information.
Kafka	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.
MapReduce	Run Terasort to generate and sort 10GB of data.
Oozie	Check that the Oozie Server Web UI page is up. Check Oozie Server status through the Oozie admin CLI shell.
Pig	Run a simple Pig script that parses a file on HDFS.
Ranger	Access administration console, view status.
Slider (See Steps 5 and 6 in "Installing Apache Slider")	Check Slider version from the Slider command line.
Sqoop	Check the Sqoop version through the Sqoop CLI shell.
Storm	Check status of the Storm Supervisor daemon from the Storm CLI. Check that the Storm UI Server web page is available.
Tez	Run Wordcount using the Tez execution engine.

7. Appendix B

Frequently-used HDP component packages:

```
hadoop_2_2_9_0_$BUILD # installs hadoop, ranger, zookeeper, hdp-
select

hadoop_2_2_9_0_$BUILD-hdfs

hadoop_2_2_9_0_$BUILD-libhdfs

hadoop_2_2_9_0_$BUILD-yarn

hadoop_2_2_9_0_$BUILD-mapreduce

hadoop_2_2_9_0_$BUILD-client

hadoop_2_2_9_0_$BUILD-httpfs

zookeeper_2_2_9_0_$BUILD # installed with hadoop

hbase_2_2_9_0_$BUILD

oozie_2_2_9_0_$BUILD # includes oozie-client

pig_2_2_9_0_$BUILD

sqoop_2_2_9_0_$BUILD

hive_2_2_9_0_$BUILD

hive_2_2_9_0_$BUILD-hcatalog

hive_2_2_9_0_$BUILD-hcatalog-server

hive_2_2_9_0_$BUILD-metastore

hive_2_2_9_0_$BUILD-server

hive_2_2_9_0_$BUILD-server2

hive_2_2_9_0_$BUILD-webhcat

hive_2_2_9_0_$BUILD-webhcat-server

hive_2_2_9_0_$BUILD-jdbc # installed with hive

tez_2_2_9_0_$BUILD # installed with hive

storm_2_2_9_0_$BUILD

falcon_2_2_9_0_$BUILD

flume_2_2_9_0_$BUILD
```

```
phoenix_2_2_9_0_$BUILD

accumulo_2_2_9_0_$BUILD

mahout_2_2_9_0_$BUILD

knox_2_2_9_0_$BUILD

ranger_2_2_9_0_$BUILD-admin

ranger_2_2_9_0_$BUILD-usersync

ranger_2_2_9_0_$BUILD-hive-plugin # installed with hive

ranger_2_2_9_0_$BUILD-storm-plugin # installed with storm

ranger_2_2_9_0_$BUILD-knox-plugin # installed with knox

ranger_2_2_9_0_$BUILD-hbase-plugin # installed with hbase

slider_2_2_9_0_$BUILD

kafka_2_2_9_0_$BUILD
```