Hortonworks Data Platform

Ambari Upgrade Guide

(Apr 22, 2014)

docs.hortonworks.com

Hortonworks Data Platform: Ambari 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. Upgrading Ambari Server to 1.5.1

This procedure upgrades Ambari Server from version 1.2.5 and above to 1.5.1. If your current Ambari Server version is 1.2.4 or below, you must upgrade the Ambari Server version to 1.2.5 before upgrading to version 1.5.1. Upgrading the Ambari Server version does not change the underlying Hadoop Stack.

Note

You must know the location of the Nagios server for Step 9. Use the **Services View**-> **Summary** panel to locate the host on which it is running.

1. Stop the Nagios and Ganglia services. In **Ambari Web**:

   a. Browse to **Services** and select each service.

   b. Use **Service Actions** to stop the service.

2. Stop the Ambari Server and all Ambari Agents. From the Ambari Server host:

   ```
   ambari-server stop
   ```

   From each Ambari Agent host:

   ```
   ambari-agent stop
   ```

3. Get the new Ambari bits. Use `wget` to fetch the repository file and replace the old repository file with the new repository file on every host.

   - Fetch the new repository file:

     For RHEL/CentOS 5/Oracle Linux 5

     ```
     wget http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/1.5.
     1/ambari.repo
     ```

     For RHEL/CentOS 6/Oracle Linux 6

     ```
     wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.5.
     1/ambari.repo
     ```

     For SLES 11

     ```
     wget http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.5.1/
     ambari.repo
     ```

     #### Important

     Check your current directory before you download the new repository file to make sure that there are no previous versions of the file. If you do not, and a previous version exists, the new download will be saved with a numeric extension such as `ambari.repo.1`. Make sure that the version you copy is the new version.

   - Replace the old repository file with the new repository file.

For RHEL/CentOS 5/Oracle Linux 5

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

For RHEL/CentOS 6/Oracle Linux 6

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

For SLES 11

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

Note

If your cluster does not have access to the Internet, set up a local repository with this data before you continue. See Configure the Local Repositories for more information.

4. Upgrade Ambari Server.

Note

Ambari Server no longer automatically turns `iptables` off. Check your installation setup to make sure that you are not relying on this function. After you have upgraded the server you must either disable iptables manually or make sure that you have all the appropriate ports available.

From the Ambari Server host:

- RHEL/CentOS/Oracle Linux

```
yum clean all
yum upgrade ambari-server ambari-log4j
```

- SLES

```
zypper clean
zypper up ambari-server ambari-log4j
```

5. Check for upgrade success:

- As the process runs, the console should produce output similar, although not identical, to this:

```
Setting up Upgrade Process
Resolving Dependencies
--> Running transaction check
---> Package ambari-server.x86_64 0:1.2.2.3-1 will be updated
---> Package ambari-server.x86_64 0:1.2.2.4-1 will be updated ...
---> Package ambari-server.x86_64 0:1.2.2.5-1 will be an update ...
```

- If the upgrade fails, the console should produce output similar to this:

```
Setting up Upgrade Process
No Packages marked for Update
```

6. Check to see if you have a folder named `/etc/ambari-server/conf.save`. If you do, rename it back:

   ```
   mv /etc/ambari-server/conf.save /etc/ambari-server/conf
   ```

7. Upgrade the Ambari Server schema. From the Ambari Server host:

   ```
   ambari-server upgrade
   ```

8. Upgrade the Ambari Agent on all hosts. From each Ambari Agent host:

   - RHEL/CentOS/Oracle Linux

     ```
     yum upgrade ambari-agent ambari-log4j
     ```

   - SLES

     ```
     zypper up ambari-agent ambari-log4j
     ```

   Note

   Ignore the warning that begins with "There are some running programs that use files deleted by recent upgrade".

9. Check to see if you have a file named `/etc/ambari-agent/conf.save` on each Agent host. If you do, rename it back. On each Agent host:

   ```
   mv /etc/ambari-agent/conf.save /etc/ambari-agent/conf
   ```

10. Upgrade the Nagios add-ons package. On the Nagios host:

    - RHEL/CentOS/Oracle Linux

      ```
      yum upgrade hdp_mon_nagios_addons
      ```

    - SLES

      ```
      zypper up hdp_mon_nagios_addons
      ```

11. After the process is complete, check each host to make sure the new 1.5.1 files have been installed:

    ```
    rpm -qa | grep ambari
    ```

12. Start the Server and the Agents on all hosts. From the Server host:

    ```
    ambari-server start
    ```

    From each Agent host:

    ```
    ambari-agent start
    ```

13. Open **Ambari Web**. Point your browser to `http://{your.ambari.server}:8080`

    Important

    Refresh your browser so that it loads the new version of the code. Hold the Shift key down while clicking the refresh button on the browser. If you have problems, clear your browser cache manually and restart Ambari Server.

Use the Admin name and password you have set up to log in.

14. Start the Nagios and Ganglia services. In **Ambari Web**:

    a. Browse to **Services** and select each service.

    b. Use **Service Actions** to start the service.

15. If you have customized logging properties, you will see refresh indicators next to each service name after upgrading to Ambari 1.5.1.

    Note

    Restarting a service pushes the configuration properties displayed in Custom log4j.properties to each host running components for that service.

    To preserve any custom logging properties after upgrading, for each service:

    a. Replace default logging properties with your custom logging properties, using Service Configs -> Custom log4j.properties.

    b. Restart all components in any services for which you have customized logging properties.

       For more information about logging properties, see Customizing Your Logging Properties.

2. Upgrading the HDP Stack from 2.0 to 2.1

The HDP Stack is the coordinated set of Hadoop components that you have installed on hosts in your cluster. Your set of Hadoop components and hosts is unique to your cluster. Before upgrading the Stack on your cluster, review all Hadoop services and hosts in your cluster to confirm the location of Hadoop components. For example, use the **Hosts** and **Services** views in the Ambari Web UI, which summarize and list the components installed on each Ambari host, to determine the components installed on each host. For more information about using Ambari to view components in your cluster, see Working with Hosts, and Viewing Components on a Host.

Complete the following procedures to upgrade the Stack from version 2.0 to version 2.1 on your current, Ambari-installed-and-managed cluster.

1. Upgrade Ambari Server to 1.5.1
2. Optional: Upgrade your existing JDK
3. Prepare HDFS for Upgrade
4. Upgrade the Stack
5. Complete the Stack Upgrade

Note

If you plan to upgrade your existing JDK, do so after upgrading Ambari, before upgrading the Stack.

2.1. Prepare the Stack for Upgrade

Perform steps 1 through 8 on the NameNode host. In a HA NameNode configuration, you should execute these on the primary NameNode. The primary NameNode is the first NameNode configured in hdfs-site.xml.

1. IF the Oozie service is installed in your cluster, list all current jobs.

```
oozie jobs -oozie http://localhost:11000/oozie -len 100 -filter status=
RUNNING
```

2. Stop all jobs in a RUNNING or SUSPENDED state on your Oozie server host. For example:

```
oozie job -oozie {your.oozie.server.host}:11000/oozie -kill {oozie.job.id}
```

3. Use the **Services** view on the **Ambari Web** UI to stop all services except HDFS and ZooKeeper. Also stop any client programs that access HDFS.

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

```
su {HDFS_USER}
hdfs dfsadmin -finalizeUpgrade
```

You can check the namenode directory to ensure that there is no snapshot of any prior HDFS upgrade. In particular, look into the directory $dfs.namenode.name.dir (or $dfs.name.dir) on the NameNode. Make sure there is only a 'current' directory and no 'previous' directory there.

5. Create the following logs and other files.

   Creating these logs allows you to check the integrity of the file system post upgrade.

   a. Run `fsck` with the following flags and send the results to a log. The resulting file contains a complete block map of the file system. You use this log later to confirm the upgrade.

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

   b. Optional: Capture the complete namespace of the file system. The following command does a recursive listing of the root file system.

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

   c. Create a list of all the DataNodes in the cluster.

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

   d. Optional: Copy all unrecoverable data stored in HDFS to a local file system or to a backup instance of HDFS.

6. **Save the namespace**. You must be the HDFS service user to do this and you must put the cluster in Safe Mode.

   ```
   hdfs dfsadmin -safemode enter
   hdfs dfsadmin -saveNamespace
   ```

   ### Note

   In a HA NameNode configuration, the command `hdfs dfsadmin -saveNamespace` does checkpoint in the first NameNode specified in the configuration, in `dfs.ha.namenodes.[nameservice ID]`. You can also use the dfsadmin `-fs` option to specify which NameNode to connect. For example, to force a checkpoint in namenode 2:

   ```
   hdfs dfsadmin -fs hdfs://namenode2-hostname:namenode2-port -
   saveNamespace
   ```

7. Copy the following checkpoint files into a backup directory. You can find the directory by using the **Services View** in the UI. Select the **HDFS** service, the **Configs** tab, in the Namenode section, look up the property **NameNode Directories**. It will be on your primary NameNode host.

   ```
   $dfs.name.dir/current
   ```

Note

In a HA NameNode configuration, the location of the checkpoint depends on where the saveNamespace command is sent, as defined in the preceding step.

8. Store the layoutVersion for the NameNode. Make a copy of the file at *`$dfs.name.dir`*`/current/VERSION` where *`$dfs.name.dir`* is the value of the config parameter `NameNode directories`. This file will be used later to verify that the layout version is upgraded.

9. Stop HDFS. Make sure all services in the cluster are completely stopped.

10. If you are upgrading Hive and Oozie, back up the Hive and Oozie metastore databases on the Hive and Oozie database host machines, respectively.

    a. Optional - Back up the Hive Metastore database.

    ### Note

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

    Table 2.1. Hive Metastore Database Backup and Rstore

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

    b. Optional - Back up the Oozie Metastore database.

    ### Note

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

    Table 2.2. Oozie Metastore Database Backup and Restore

Database Type	Backup	Restore
MySQL	`mysqldump $dbname > $outputfilename.sql` For	`mysql $dbname < $inputfilename.sql` For

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

11. Stop Ambari Server and confirm that it is stopped. On the Server host:

```
ambari-server stop
```

```
ambari-server status
```

12. Stop all Ambari Agents. On each host:

```
ambari-agent stop
```

2.2. Upgrade the Stack

1. Upgrade the HDP repository on all hosts and replace the old repository file with the new file:

   • For RHEL/CentOS/Oracle Linux 5

   ```
   wget -O /etc/yum.repos.d/HDP.repo http://public-repo-1.hortonworks.com/
   HDP/centos5/2.x/GA/2.1-latest/hdp.repo
   ```

   • For RHEL/CentOS/Oracle Linux 6

   ```
   wget -O /etc/yum.repos.d/HDP.repo http://public-repo-1.hortonworks.com/
   HDP/centos6/2.x/GA/2.1-latest/hdp.repo
   ```

   • For SLES 11

   ```
   wget -O /etc/yum.repos.d/HDP.repo http://public-repo-1.hortonworks.com/
   HDP/suse11/2.x/GA/2.1-latest/hdp.repo
   ```

   Important

   Make sure to download the HDP.repo file under /etc/yum.repos.d on ALL hosts.

2. Update the stack version in the Server database. Use the command appropriate for a remote, or local repository, as described in this step.

   ```
   ambari-server upgradestack HDP-2.1 {HDP.Base.URL} {$os}
   ```

   For a remote, accessible, public repository, the `HDP.Base.URL` is the same as the `baseurl={HDP.Base.URL}` in the HDP.repo file download in Step 1. For a local repository, use the local repository Base URL you have configured for the HDP Stack. For links to download the HDP repository files for your version of the Stack, see HDP Stack Repositories.

3. Back up the files in following directories on the Oozie server host and make sure that all files, including *site.xml files are copied.

   ```
   mkdir oozie-conf-bak
   cp -R /etc/oozie/conf/* oozie-conf-bak
   ```

4. Remove the old oozie directories on all Oozie server and client hosts

   - rm -rf /etc/oozie/conf
   - rm -rf /usr/lib/oozie/
   - rm -rf /var/lib/oozie/

5. Upgrade the stack on all Agent hosts. Skip any components your installation does not use.:

   - For RHEL/CentOS/Oracle Linux

     a. Remove WebHCat, HCatalog, and Oozie components.

        ```
        yum erase "webhcat*" "hcatalog*" "oozie*"
        ```

     b. Upgrade the following components:

        ```
        yum upgrade "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
         "sqoop*" "zookeeper*" "hbase*" "hive*" hdp_mon_nagios_addons
        ```

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

        ```
        yum install hive*
        ```

        ```
        yum install oozie oozie-client
        ```

        ```
        rpm -e --nodeps bigtop-jsvc
        ```

        ```
        yum install bigtop-jsvc
        ```

     c. Verify that the components were upgraded:

        ```
        yum list installed | grep HDP-$old-stack-version-number
        ```

        None of the components from that list should appear in the returned list.

   - For SLES

     a. Remove WebHCat, HCatalog, and Oozie components.

        ```
        zypper remove webhcat\* hcatalog\* oozie\*
        ```

     b. Upgrade the following components:

        ```
        zypper up "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
         "sqoop*" "zookeeper*" "hbase*" "hive*" hdp_mon_nagios_addons
        ```

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

        ```
        zypper up -r HDP-2.1.1.0
        ```

        ```
        zypper install hive\*
        ```

```
zypper install oozie oozie-client
```

c. Verify that the components were upgraded:

```
rpm -qa | grep hadoop, rpm -qa | grep hive and rpm -qa | grep hcatalog
```

d. If components were not upgraded, upgrade them as follows:

```
yast --update hadoop hcatalog hive
```

2.3. Complete the Stack Upgrade

1. Start Ambari Server and Ambari Agents.

   On the Server host:

   ```
   ambari-server start
   ```

   On all of the Agent hosts:

   ```
   ambari-agent start
   ```

2. Using the Ambari Web Services view, start the ZooKeeper service.

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

   ```
   su -l {HDFS_USER} -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh start
    journalnode"
   ```

   ### Important

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

4. Because the file system version has now changed you must start the NameNode manually. On the active NameNode host:

   ```
    su -l {HDFS_USER} -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec &&
    /usr/lib/hadoop/sbin/hadoop-daemon.sh start namenode -upgrade"
   ```

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

   ### Note

   In a NameNode HA configuration, this NameNode will not enter the standby state as usual. Rather, this NameNode will immediately enter the active state, perform an upgrade of its local storage directories, and also perform an upgrade of the shared edit log. At this point, the standby NameNode in the HA pair is still down. It will be out of sync with the upgraded active NameNode. To synchronize the active and standby NameNode, re-establishing HA, re-bootstrap the standbyNameNode by running the NameNode with the '-bootstrapStandby' flag. **Do NOT** start this standby NameNode with the '-upgrade' flag.

```
su -l {HDFS_USER} -c "hdfs namenode -bootstrapStandby -force"
```

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

5. Start all DataNodes.

```
su -l hdfs -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh --config /etc/hadoop/
conf start datanode"
```

The NameNode will send an upgrade command to DataNodes after receiving block reports.

To check if the Upgrade is in progress, look for `\previous` in NameNode/DataNode/ JournalNode directory. `\previous` contains a snapshot of the data before upgrade.

6. Prepare the NameNode to work with Ambari:

   a. Open the Ambari Web GUI. If it has been open throughout the process, clear your browser cache, then refresh.

   b. On the Services view, choose **HDFS** to open the HDFS service.

   c. Restart the HDFS service. Restarting HDFS restarts all NameNodes, DataNodes, and JournalNodes.

   d. Run the Service Check, using Actions > Run Service Check. Makes sure it passes.

7. After the DataNodes are started, HDFS exits safemode. To monitor the status, run the following command:

```
sudo su -l {HDFS_USER} -c "hdfs dfsadmin -safemode get"
```

Depending on the size of your system, a response may not display for up to 10 minutes. When HDFS exits safemode, the following message displays:

```
Safe mode is OFF
```

8. Make sure that the HDFS upgrade was successful. Go through steps 2, 3, and 4 in Prepare HDFS for the Upgrade to create new versions of the logs and reports. Substitute "`new`" for "`old`" in the file names as necessary.

9. Compare the old and new versions of the following:

   • `dfs-old-fsck-1.log` versus `dfs-new-fsck-1.log`.

     The files should be identical unless the `hadoop fsck` reporting format has changed in the new version.

- `dfs-old-lsr-1.log` versus `dfs-new-lsr-1.log`.

  The files should be identical unless the format of `hadoop fs -lsr` reporting or the data structures have changed in the new version.

- `dfs-old-report-1.log` versus `fs-new-report-1.log`

  Make sure all DataNodes previously belonging to the cluster are up and running.

10. Update the configuration properties required for Application Timeline Server. Using Ambari Web -> Services -> Configs, choose a service, then add/modify values for each of the following properties:

```
YARN (Custom yarn-site.xml)
yarn.timeline-service.leveldb-timeline-store.path=/var/log/hadoop-yarn/
timeline
yarn.timeline-service.leveldb-timeline-store.ttl-interval-ms=300000
yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.
applicationhistoryservice.timeline.LeveldbTimelineStore
yarn.timeline-service.ttl-enable=true
yarn.timeline-service.ttl-ms=2678400000
yarn.timeline-service.generic-application-history.store-class=org.apache.
hadoop.yarn.server.applicationhistoryservice.NullApplicationHistoryStore
yarn.timeline-service.webapp.address=
{PUT_THE_FQDN_OF_RESOURCEMANAGER_HOST_NAME_HERE}:8188
yarn.timeline-service.webapp.https.address=
{PUT_THE_FQDN_OF_RESOURCEMANAGER_HOST_NAME_HERE}:8190
yarn.timeline-service.address=0.0.0.0:10200

HIVE (hive-site.xml)
hive.execution.engine=mr
hive.exec.failure.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.pre.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.tez.container.size={map-container-size}
        *If mapreduce.map.memory.mb > 2GB then set it equal to mapreduce.
map.memory.*
        *Otherwise set it equal to mapreduce.reduce.memory.mb*
```

11. Use the Ambari Web Services view to start YARN.

12. Use the Ambari Web Services view to start MapReduce2.

13. Use the Ambari Web Services view to start HBase and ensure the service check passes.

14. Upgrade Oozie.

    Note

    You must replace your Oozie configuration after upgrading.

    a. Perform the following preparation steps on each oozie server and client:

       i. Copy files from the backup folder `/conf` to `/etc/oozie/conf` directory.

```
cp {oozie-conf-bak}/oozie-site.xml /etc/oozie/conf
```

```
cp {oozie-conf-bak}/oozie-env.sh /etc/oozie/conf/oozie-env.sh
```

```
chmod -R 777 /etc/alternatives/oozie-tomcat-conf/conf
```

```
rm /usr/lib/oozie/conf
```

```
ln -s /etc/oozie/conf /usr/lib/oozie/conf
```

ii. Create `/usr/lib/oozie/libext-upgrade21` directory.

```
mkdir /usr/lib/oozie/libext-upgrade21
```

iii. Copy the JDBC jar of your Oozie database to both `/usr/lib/oozie/libext-upgrade21` and `/usr/lib/oozie/libtools`.

For example, if you are using MySQL, copy your `mysql-connector-java.jar`.

iv. Copy these files to `/usr/lib/oozie/libext-upgrade21` directory

```
cp /usr/lib/hadoop/lib/hadoop-lzo*.jar /usr/lib/oozie/libext-upgrade21
```

```
cp /usr/share/HDP-oozie/ext-2.2.zip /usr/lib/oozie/libext-upgrade21
```

v. Grant read/write access to the Oozie user.

```
chmod -R 777 /usr/lib/oozie/libext-upgrade21
```

b. Upgrade steps:

i. On the Services view, make sure YARN and MapReduce2 are running.

ii. Make sure that the Oozie service is stopped.

iii. Upgrade Oozie. You must be the Oozie service user. On the Oozie host:

```
sudo su -l {OOZIE_USER} -c"/usr/lib/oozie/bin/ooziedb.sh upgrade -run"
```

Make sure that the output contains the string "Oozie DB has been upgraded to Oozie version *`{OOZIE Build Version}`*.

iv. Prepare the Oozie WAR file, run as root:

Note

The Oozie server must be **not** running for this step. If you get the message "ERROR: Stop Oozie first", it means the script still thinks it's running. Check, and if needed, remove the process id (pid) file indicated in the output.

```
sudo su -l {OOZIE_USER} -c "/usr/lib/oozie/bin/oozie-setup.sh prepare-
war -d /usr/lib/oozie/libext-upgrade21"
```

Make sure that the output contains the string "New Oozie WAR file with added".

v. Using Ambari Web UI **Services** > **Oozie** > **Configs**, expand **Advanced**, then edit the following properties:

A. In `oozie.service.coord.push.check.requeue.interval`, ***replace*** the existing property value with the following one:

```
30000
```

B. In `oozie.service.SchemaService.wf.ext.schemas`, ***append*** (using copy/paste) to the existing property value the following string:

```
shell-action-0.2.xsd,oozie-sla-0.1.xsd,oozie-sla-0.2.xsd,hive-
action-0.3.xsd
```

Note

If you have customized schemas, append this string to your custom schema name string.

Do not overwrite custom schemas.

If you have no customized schemas, you can replace the existing string with the following one:

shell-action-0.1.xsd,email-action-0.1.xsd,hive-action-0.2.xsd, sqoop-action-0.2.xsd,ssh-action-0.1.xsd,distcp-action-0.1.xsd,shell-action-0.2.xsd,oozie-sla-0.1.xsd,oozie-sla-0.2.xsd,hive-action-0.3.xsd

C. In `oozie.service.URIHandlerService.uri.handlers`, ***append*** to the existing property value the following string:

```
org.apache.oozie.dependency.FSURIHandler,org.apache.oozie.dependency.
HCatURIHandler
```

D. In `oozie.services`, **append** to the existing property value the following string:

```
org.apache.oozie.service.XLogStreamingService,org.apache.oozie.
service.JobsConcurrencyService
```

Note

If you have customized properties, append this string to your custom property value string.

Do not overwrite custom properties.

If you have no customized properties, you can replace the existing string with the following one:

org.apache.oozie.service.SchedulerService,
org.apache.oozie.service.InstrumentationService,
org.apache.oozie.service.CallableQueueService,
org.apache.oozie.service.UUIDService,
org.apache.oozie.service.ELService,
org.apache.oozie.service.AuthorizationService,
org.apache.oozie.service.UserGroupInformationService,

org.apache.oozie.service.HadoopAccessorService,
org.apache.oozie.service.URIHandlerService,
org.apache.oozie.service.MemoryLocksService,
org.apache.oozie.service.DagXLogInfoService,
org.apache.oozie.service.SchemaService,
org.apache.oozie.service.LiteWorkflowAppService,
org.apache.oozie.service.JPAService,
org.apache.oozie.service.StoreService,
org.apache.oozie.service.CoordinatorStoreService,
org.apache.oozie.service.SLAStoreService,
org.apache.oozie.service.DBLiteWorkflowStoreService,
org.apache.oozie.service.CallbackService,
org.apache.oozie.service.ActionService,
org.apache.oozie.service.ActionCheckerService,
org.apache.oozie.service.RecoveryService,
org.apache.oozie.service.PurgeService,
org.apache.oozie.service.CoordinatorEngineService,
org.apache.oozie.service.BundleEngineService,
org.apache.oozie.service.DagEngineService,
org.apache.oozie.service.CoordMaterializeTriggerService,
org.apache.oozie.service.StatusTransitService,
org.apache.oozie.service.PauseTransitService,
org.apache.oozie.service.GroupsService,
org.apache.oozie.service.ProxyUserService,
org.apache.oozie.service.XLogStreamingService,
org.apache.oozie.service.JobsConcurrencyService

E. In `oozie.services.ext`, ***append*** to the existing property value the following string:

```
org.apache.oozie.service.PartitionDependencyManagerService,org.
apache.oozie.service.HCatAccessorService
```

F. After modifying all properties on the Oozie Configs page, scroll down, then choose **Save** to update oozie.site.xml, using the modified configurations.

vi. Replace the content of `/usr/oozie/share` in HDFS. On the Oozie server host:

A. Extract the Oozie sharelib into a `tmp` folder.

```
mkdir -p /tmp/oozie_tmp
cp /usr/lib/oozie/oozie-sharelib.tar.gz /tmp/oozie_tmp
cd /tmp/oozie_tmp
tar xzvf oozie-sharelib.tar.gz
```

B. Back up the`/user/oozie/share` folder in HDFS and then delete it. If you have any custom files in this folder back them up separately and then add them back after the share folder is updated.

```
mkdir /tmp/oozie_tmp/oozie_share_backup
chmod 777 /tmp/oozie_tmp/oozie_share_backup
```

```
su -l {HDFS_USER} -c "hdfs dfs -copyToLocal /user/oozie/share /tmp/
oozie_tmp/oozie_share_backup"
su -l {HDFS_USER} -c "hdfs dfs -rm -r /user/oozie/share"
```

C. Add the latest share libs that you extracted in step 1. After you have added the files, modify ownership and acl.

```
su -l {HDFS_USER} -c "hdfs dfs -copyFromLocal /tmp/oozie_tmp/share /
user/oozie/."
su -l {HDFS_USER} -c "hdfs dfs -chown -R {OOZIE_USER}:
{HADOOP_GROUP} /user/oozie"
su -l {HDFS_USER} -c "hdfs dfs -chmod -R 755 /user/oozie"
```

vii. Use the Services view to start the Oozie service. Make sure that ServiceCheck passes for Oozie.

15. Update WebHCat.

a. Modify the `webhcat-site` config type.

Using the Ambari web UI, navigate to **Services > WebHCat** and modify the following configuration:

Table 2.3. WebHCat Properties to Modify

Action	Property Name	Property Value
Modify	templeton.storage.class	`org.apache.hive.hcatalog.templeto`

b. Update the Pig and Hive tar bundles, by updating the following files:

- /apps/webhcat/pig.tar.gz
- /apps/webhcat/hive.tar.gz

Note

You will find these files on a host where webhcat is installed.

For example, to update a *.tar.gz file:

i. Move the file to a local directory.

```
su -l {HCAT_USER} -c "hadoop --config /etc/hadoop/conf fs -
copyToLocal /apps/webhcat/*.tar.gz ${local_backup_dir}"
```

ii. Remove the old file.

```
su -l {HCAT_USER} -c "hadoop --config /etc/hadoop/conf fs -rm /apps/
webhcat/*.tar.gz"
```

iii. Copy the new file.

```
su -l {HCAT_USER} -c "hdfs --config /etc/hadoop/conf dfs -
copyFromLocal /usr/share/HDP-webhcat/*.tar.gz /apps/webhcat/"
```

c. Update /app/webhcat/hadoop-streaming.jar file.

i. Move the file to a local directory.

```
su -l {HCAT_USER} -c "hadoop --config /etc/hadoop/conf fs -
copyToLocal /apps/webhcat/hadoop-streaming*.jar ${local_backup_dir}"
```

ii. Remove the old file.

```
su -l {HCAT_USER} -c "hadoop --config /etc/hadoop/conf fs -rm /apps/
webhcat/hadoop-streaming*.jar"
```

iii. Copy the new hadoop-streaming.jar file.

```
su -l {HCAT_USER} -c "hdfs --config /etc/hadoop/conf dfs -
copyFromLocal /usr/lib/hadoop-mapreduce/hadoop-streaming*.jar /apps/
webhcat"
```

16.Upgrade Flume.

a. Make a backup copy of the current Flume configuration files, on each Flume host.

```
cd /etc/flume/conf
cp flume.conf {flume-conf-backup}/flume.conf
```

Note

More than one Flume configuration file may exist. Make a backup copy of each one.

b. Execute the following commands on each Flume host:

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

```
yum upgrade flume
```

- For **SLES**:

```
zypper upgrade flume
zypper remove flume
zypper se -s flume
```

You should see Flume in the output. Then, install Flume.

```
zypper install flume
```

Important

When removing and installing packages, rename files in the `/conf` directory that have `.rpmsave` extension to original to retain the customized configurations. Alternatively, use the configuration files in the `/conf` directory that you backed up before upgrading.

c. Verify that Flume was upgraded correctly by starting a basic example. Flume does not start running immediately after installation/upgrade, by default. To validate the Flume upgrade:

i. Replace your default `conf/flume.conf` with the following `flume.conf` file:

```
1. Name the components on this agent
```

```
    a1.sources = r1
    a1.sinks = k1
    a1.channels = c1
2.Describe/configure the source
    a1.sources.r1.type = seq
3. Describe the sink
    a1.sinks.k1.type = file_roll
    a1.sinks.k1.channel = c1
    a1.sinks.k1.sink.directory = /tmp/flume
4. Use a channel which buffers events in memory
    a1.channels.c1.type = memory
5. Bind the source and sink to the channel
    a1.sources.r1.channels = c1
    a1.sinks.k1.channel = c1
```

ii. Restart Flume, using Ambari Web.

iii. To verify that data is flowing, examine `/tmp/flume` to see that any files exist. These files should contain simple, sequential numbers.

iv. Stop Flume.

d. Copy the backup Flume configuration files you created in Step 17.a back into their original location.

```
cd /etc/flume/conf
cp {flume-conf-backup}/flume.conf
```

17.Using **Ambari Web > Services**, re-start the remaining services.

18.The upgrade is now fully functional but not yet finalized. Using the `finalize` command removes the previous version of the NameNode and DataNode storage directories.

Important

After the upgrade is finalized, the system cannot be rolled back. Usually this step is not taken until a thorough testing of the upgrade has been performed.

The upgrade must be finalized before another upgrade can be performed.

Note

Directories used by Hadoop 1 services set in /etc/hadoop/conf/taskcontroller.cfg are not automatically deleted after upgrade. Administrators can choose to delete these directories after the upgrade.

To finalize the upgrade:

```
sudo su -l {HDFS_USER} -c "hdfs dfsadmin -finalizeUpgrade"
```

3. Upgrading the HDP Stack from 1.3.2 or later to 2.1

The stack is the coordinated set of Hadoop components that you have installed. Use the following instructions to 1) upgrade a current, Ambari-installed and managed instance of a 1.3.2 or later version stack to a 2.1 version of the stack and 2) upgrade Ambari Server and Agents to 1.5.1 version. This procedure causes the upgraded stack to be managed by Ambari.

Note

If your stack has Kerberos Security turned on, you should turn it off before performing the upgrade. On Ambari Web->Admin view->Security-> click **Disable Security**. You can re-enable Security after performing the upgrade.

If you are upgrading from any other 1.x version of the stack, you must upgrade to 1.3.2 or later before you can upgrade to 2.1. See Upgrading the HDP Stack to 1.3.3 for more information. Upgrades from previous 1.x versions are not supported.

Note

If you have already upgraded to Ambari Server 1.5.1 and just want to upgrade the HDP stack, skip Section 2 and Section 3 and go directly to Section 4.

3.1. Preparing for the Upgrade

Use the following steps to prepare your system for the upgrade.

1. If you are upgrading Ambari as well as the stack, you must know the location of the Nagios servers for that process. Use the **Services**->**Nagios**-> **Summary** panel to locate the hosts on which they are running.

2. IF the Oozie service is installed in your cluster, list all current jobs.

   ```
   oozie jobs -oozie http://localhost:11000/oozie -len 100 -filter status=
   RUNNING
   ```

3. Stop all jobs in a RUNNING or SUSPENDED state on your Oozie server host. For example:

   ```
   oozie job -oozie {your.oozie.server.host}:11000/oozie -kill {oozie.job.id}
   ```

4. Use the **Services** view on the **Ambari Web** UI to stop all services, including all clients, running on HDFS. Do **not** stop HDFS yet.

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

   ```
   su $HDFSUSER
   hadoop namenode -finalize
   ```

6. Create the following logs and other files.

Because the upgrade to 2.1 includes a version upgrade of HDFS, creating these logs allows you to check the integrity of the file system post upgrade.

a. Run `fsck` with the following flags and send the results to a log. The resulting file contains a complete block map of the file system. You use this log later to confirm the upgrade.

```
su $HDFS_USER
hadoop fsck / -files -blocks -locations > /tmp/dfs-old-fsck-1.log
```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

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

```
su $HDFS_USER
hadoop dfs -lsr / > /tmp/dfs-old-lsr-1.log
```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

c. Create a list of all the DataNodes in the cluster.

```
su $HDFS_USER
hadoop dfsadmin -report > /tmp/dfs-old-report-1.log
```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

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

e. Optional: create the logs again and check to make sure the results are identical.

7. Save the namespace. You must be the HDFS service user to do this and you must put the cluster in Safe Mode.

Important

This is a **critical** step. If you do not do this step before you do the upgrade, the NameNode will **not** start afterwards.

```
su $HDFS_USER
hadoop dfsadmin -safemode enter
hadoop dfsadmin -saveNamespace
```

8. Copy the following checkpoint files into a backup directory. You can find the directory by using the **Services View** in the UI. Select the **HDFS** service, the **Configs** tab, in the Namenode section, look up the property **NameNode Directories**. It will be on your NameNode host.

- `dfs.name.dir/edits`
- `dfs.name.dir/image/fsimage`
- `dfs.name.dir/current/fsimage`

9. On the JobTracker host, copy `/etc/hadoop/conf` to a backup directory.

Note

If you have deployed a custom version of `capacity-scheduler.xml` and `mapred-queue-acls.xml`, after the upgrade you will need to use Ambari Web to edit the default Capacity Scheduler. Select Services view ->**YARN**->**Configs**->**Scheduler**->**Capacity Scheduler**.

Important

Fair Scheduler is not supported for use with HDP 2.x.

10. Store the layoutVersion for the NameNode. Make a copy of the file at *`$dfs.name.dir`*`/current/VERSION` where *`$dfs.name.dir`* is the value of the config parameter `NameNode directories`. This file will be used later to verify that the layout version is upgraded.

11. Stop HDFS. Make sure all services in the cluster are completely stopped.

12. If you are upgrading Hive and Oozie, back up the Hive database and the Oozie database.

13. Stop Ambari Server. On the Server host:

    ```
    ambari-server stop
    ```

14. Stop Ambari Agents. On each host:

    ```
    ambari-agent stop
    ```

3.2. Setting Up the Ambari Repository

This process prepares the updated repository.

1. Check to see if you have a `conf.save` directory for Ambari server and agents. If you do, move them to a back-up location:

   ```
   mv /etc/ambari-server/conf.save/ /etc/ambari-server/conf.save.bak
   ```

   ```
   mv /etc/ambari-agent/conf.save/ /etc/ambari-agent/conf.save.bak
   ```

2. Get the new Ambari bits. Use `wget` to fetch the repository file and replace the old repo file with the new repo file on every host.

   Important

   Check your current directory before you download the new repo file to make sure that there are no previous versions of the file. If you do not, and a previous version exists, the new download is saved with a numeric extension such as `ambari.repo.1`. Make sure that the version you copy is the new version.

   - For RHEL/CentOS/Oracle Linux 5

```
wget http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/1.5.
1/ambari.repo
cp ambari.repo /etc/yum.repos.d/ambari.repo
```

- For RHEL/CentOS/Oracle Linux 6

```
wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.5.
1/ambari.repo
cp ambari.repo /etc/yum.repos.d/ambari.repo
```

- For SLES 11

```
wget http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.5.1/
ambari.repo
cp ambari.repo /etc/zypp/repos.d/ambari.repo
```

Note

If your cluster does not have access to the Internet, you need to set up a local repository with this data before you continue. See Configure the Local Repositories for more information.

3.3. Upgrading to Ambari 1.5.1

This process upgrades Ambari Server, Ambari Agents, Ganglia, and Nagios.

Note

Ambari Server no longer automatically turns `iptables` off. Check your installation setup to make sure that you are not relying on this function. After you have upgraded the server you must either disable iptables manually or make sure that you have all the appropriate ports available.

1. Upgrade Ambari Server. From the Ambari Server host:

   - RHEL/CentOS/Oracle Linux

   ```
   yum clean all
   yum upgrade ambari-server ambari-log4j
   ```

   - SLES

   ```
   zypper clean
   zypper up ambari-server ambari-log4j
   ```

2. Check for upgrade success:

   - As the process runs, the console should produce output similar, although not identical, to this:

   ```
   Setting up Upgrade Process
   Resolving Dependencies
   --> Running transaction check
   ---> Package ambari-server.x86_64 0:1.2.2.3-1 will be updated
   ---> Package ambari-server.x86_64 0:1.2.2.4-1 will be updated ...
   ```

```
---> Package ambari-server.x86_64 0:1.2.2.5-1 will be an update ...
```

After the process is complete, check each host to make sure the new 1.5.1 files have been installed:

```
rpm -qa | grep ambari
```

- If the upgrade fails, the console should produce output similar to this:

```
Setting up Upgrade Process
No Packages marked for Update
```

3. Check to see if you have a folder named `/etc/ambari-server/conf.save`. If you do, rename it back:

```
mv /etc/ambari-server/conf.save /etc/ambari-server/conf
```

4. Upgrade the Ambari Server schema. From the Ambari Server host:

```
ambari-server upgrade
```

5. Upgrade the Ambari Agent on all hosts. From each Ambari Agent host:

   - RHEL/CentOS/Oracle Linux

   ```
   yum upgrade ambari-agent ambari-log4j
   ```

   - SLES

   ```
   zypper up ambari-agent ambari-log4j
   ```

   Note

   If you get a warning that begins "There are some running programs that use files deleted by recent upgrade," you can ignore it.

6. Check to see if you have a folder named `/etc/ambari-agent/conf.save` on each Agent host. If you do, rename it back. On each Agent host:

```
mv /etc/ambari-agent/conf.save /etc/ambari-agent/conf
```

7. Upgrade the Nagios addons:

   - RHEL/CentOS/Oracle Linux

   ```
   yum upgrade hdp_mon_nagios_addons
   ```

   - SLES

   ```
   zypper up hdp_mon_nagios_addons
   ```

3.4. Upgrading the Stack

This stack upgrade involves removing the HDP 1.x version of MapReduce and replacing it with the HDP 2.x YARN and MapReduce2 components. This process is somewhat long and complex. To help you, a Python script is provided to automate some of the upgrade steps.

3.4.1. Prepare for the Stack Upgrade

1. Make sure that you completed the system preparation procedure; most importantly, save the namespace.

2. Stage the upgrade script:

   a. Create an "Upgrade Folder", for example `/work/upgrade_hdp_2`, on a host that can communicate with Ambari Server. The Ambari Server host would be a suitable candidate.

   b. Copy the upgrade script to the Upgrade Folder. The script is available here: `/var/lib/ambari-server/resources/scripts/UpgradeHelper_HDP2.py` on the Ambari Server host.

   c. Make sure that Python is available on the host and that the version is 2.6 or higher:

   ```
   python --version
   ```

   Note

   For RHEL/Centos/Oracle Linux 5, you **must** use Python 2.6.

3. Start Ambari Server only. On the Ambari Server host:

   ```
   ambari-server start
   ```

4. Back up current configuration settings and the component host mappings from MapReduce:

   a. Go to the Upgrade Folder.

   b. Execute the `backup-configs` action:

   ```
   python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
   password $PASSWORD --clustername $CLUSTERNAME backup-configs
   ```

   Where

   - *$HOSTNAME* is the name of the Ambari Server host
   - *$USERNAME* is the admin user for Ambari Server
   - *$PASSWORD* is the password for the admin user
   - *$CLUSTERNAME* is the name of the cluster

   This step produces a set of files named TYPE_TAG, where TYPE is the configuration type and TAG is the tag. These files contain copies of the various configuration settings for the current (pre-upgrade) cluster. You can use these files as a reference later.

   c. Execute the `save-mr-mapping` action:

   ```
   python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
   password $PASSWORD --clustername $CLUSTERNAME save-mr-mapping
   ```

This step produces a file named `mr_mapping` that stores the host level mapping of MapReduce components such as MapReduce JobTracker/TaskTracker/Client.

5. Delete all the MapReduce server components installed on the cluster.

   a. If you are not already there, go to the Upgrade Folder.

   b. Execute the `delete-mr` action.

   ```
   python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
   password $PASSWORD --clustername $CLUSTERNAME delete-mr
   ```

   Optionally, execute the delete script with the -n option to view, verify, and validate API calls, if necessary.

   Note

   Running the delete script with the -n option exposes API calls but does not remove installed components. Use the -n option for validation purposes only.

   c. The script asks you to confirm that you have executed the `save-mr-mapping` action and that you have a file named `mr_mapping` in the Upgrade Folder.

3.4.2. Upgrade the Stack

1. Stop Ambari Server. On the Ambari Server host:

   ```
   ambari-server stop
   ```

2. Update the stack version in the Server database. Use the command appropriate for a remote, or local repository, as described in this step.

   Important

   Make sure you delete the old MapReduce version **before** you run `upgradestack`.

   ```
   ambari-server upgradestack HDP-2.1
   ```

   Note

   When upgrading the Stack, you may use a local repository. To upgrade the Stack using a local repository, provide the local HDP repo URL and target OS type as parameters in the upgradestack command, as shown in the following example:

   ```
   $ ambari-server upgradestack HDP-2.1 {HDP.Base.URL} {$os}
   ```

   For more infomation about upgrading from a local repository, see Setting Up a Local Repository.

3. Upgrade the HDP repository on all hosts and replace the old repo file with the new file:

Important

The file you download is named `hdp.repo`. To function properly in the system, it must be named `HDP.repo`. Once you have completed the "mv" of the new repo file to the repos.d folder, make sure there is no file named `hdp.repo` anywhere in your repos.d folder.

- For RHEL/CentOS/Oracle Linux 5

  ```
  wget  http://public-repo-1.hortonworks.com/HDP/centos5/2.x/GA/2.1-latest/
  hdp.repo /etc/yum.repos.d/HDP.repo
  ```

- For RHEL/CentOS/Oracle Linux 6

  ```
  wget http://public-repo-1.hortonworks.com/HDP/centos6/2.x/GA/2.1-latest/
  hdp.repo /etc/yum.repos.d/HDP.repo
  ```

- For SLES 11

  ```
  wget  http://public-repo-1.hortonworks.com/HDP/suse11/2.x/GA/2.1-latest/
  hdp.repo /etc/zypp/repos.d/HDP.repo
  ```

4. Back up the files in following directories to a tmp folder:
   - /etc/webhcat/conf
   - /etc/oozie/conf
5. Remove the old oozie directories
   - rm -rf /etc/oozie/conf
   - rm -rf /var/lib/oozie/conf
   - rm -rf /var/lib/oozie/oozie-server/conf
6. Upgrade the stack on all Agent hosts.

Note

For each host, identify the HDP components installed on each host. Use Ambari Web, as described here, to view components on each host in your cluster. Based on the HDP components installed, tailor the following upgrade commands for each host to upgrade only components residing on that host. For example, if you know that a host has **no** HBase service or client packages installed, then you can adapt the command to **not** include HBase, as follows:

```
yum upgrade "collectd*" "gccxml*" "pig*" "hadoop*" "sqoop*"
 "zookeeper*" "hive*"
```

- For RHEL/CentOS/Oracle Linux

  a. Remove remaining MapReduce components on all hosts:

  ```
  yum erase hadoop-pipes hadoop-sbin hadoop-native
  ```

```
yum erase "webhcat*" "hcatalog*" "oozie*"
```

b. Upgrade the following components:

```
yum upgrade "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
 "sqoop*" "zookeeper*" "hbase*" "hive*" hdp_mon_nagios_addons
```

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

```
yum install hive*
```

```
yum install oozie oozie-client
```

```
rpm -e --nodeps bigtop-jsvc
```

```
yum install bigtop-jsvc
```

c. Verify that the components were upgraded:

```
yum list installed | grep HDP-$old-stack-version-number
```

None of the components from that list should appear in the returned list.

- For SLES

a. Remove remaining MapReduce components on all hosts:

```
zypper remove hadoop-pipes hadoop-sbin hadoop-native
```

```
zypper remove webhcat\* hcatalog\* oozie\*
```

b. Upgrade the following components:

```
zypper up "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
 "sqoop*" "zookeeper*" "hbase*" "hive*" hdp_mon_nagios_addons
```

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

```
zypper up -r HDP-2.1.1.0
```

```
zypper install hive\*
```

```
zypper install oozie oozie-client
```

c. Verify that the components were upgraded:

```
rpm -qa | grep hadoop, rpm -qa | grep hive and rpm -qa | grep hcatalog
```

d. If components were not upgraded, upgrade them as follows:

```
yast --update hadoop hcatalog hive
```

3.4.3. Add YARN/MR2 and Update Configurations

1. Start the Ambari Server. On the Server host:

```
ambari-server start
```

2. Start each Ambari Agent. On all Agent hosts:

```
ambari-agent start
```

3. After the Server and all Agents are running, log into Ambari Web. Do a hard refresh on your browser to make sure you are displaying the updated GUI. Make sure all hosts are healthy and all services are in a Stopped state.

4. Add YARN and MapReduce2 services:

   a. If you are not already there, go to the Upgrade Folder.

   b. Execute the `add-yarn-mr2` action:

   ```
   python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
   password $PASSWORD --clustername $CLUSTERNAME add-yarn-mr2
   ```

   If desired, you can use the -n option to see the API calls as they are being made so that you can verify them.

5. Update the respective configurations:

   a. If you are not already there, go to the Upgrade Folder.

   b. Execute the `update-configs` action:

   ```
   python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
   password $PASSWORD --clustername $CLUSTERNAME update-configs
   ```

6. Update individual configuration settings as needed. On the Ambari Server, use `/var/lib/ambari-server/resources/scripts/configs.sh` to inspect and update the configuration properties.

   Note

   configs.sh creates temporary files. We recommended that you run configs.sh as root or as a user having write permission on the local folder.

   a. Get configuration details:

   ```
   configs.sh get $HOSTNAME $CLUSTERNAME $CONFIGURATION-TYPE

   For example:
   configs.sh get localhost myclustername global
   ```

   b. Evaluate each property value returned and modify as needed:

   ```
   configs.sh set $HOSTNAME $CLUSTERNAME $CONFIGURATION-TYPE "property name"
    "new value"

   For example:
   configs.sh set localhost myclustername global yarn_log_dir_prefix "/apps/
   logs/yarn"
   ```

   c. Remove properties that are not needed:

   ```
   configs.sh delete $HOSTNAME $CLUSTERNAME $CONFIGURATION-TYPE "property
    name"
   ```

```
For example:
configs.sh delete localhost myclustername global dfs.client-write-packet-
size
```

Table 3.1. Key Properties to Check

Configuration Type	Property	Description
global	yarn_log_dir_prefix	The location for the YARN logs
global	yarn_pid_dir_prefix	The location for the YARN pid files
global	yarn_user	The YARN user
yarn-site	yarn.nodemanager.local-dirs	The location for container logs
yarn-site	yarn.nodemanager.log-dirs	The directories for localized files

Note

Make any necessary modifications **before** starting the services.

d. Install the YARN and MapReduce2 services:

i. If you are not already there, go to the Upgrade Folder.

ii. Execute the `install-yarn-mr2` action:

```
python UpgradeHelper_HDP2.py --hostname $HOSTNAME --user $USERNAME --
password $PASSWORD --clustername $CLUSTERNAME install-yarn-mr2
```

Note

This is a two-step process. You can use the Ambari Web GUI to monitor the progress. Both steps must be **complete** before you continue to the next step.

3.4.4. Complete the Stack Upgrade

1. Start Ambari Server and Ambari Agents.

   On the Server host:

   ```
   ambari-server start
   ```

   On all of the Agent hosts:

   ```
   ambari-agent start
   ```

2. Update the repository Base URLs in Ambari Server for the HDP-2.1 stack. Browse to **Ambari Web > Admin > Clusters** and set the value of the HDP and HDP-UTILS repository Base URLs. For more information about viewing and editing repository Base URLs, see Managing Stack Repositories.

   Important

   For a remote, accessible, public repository, the HDP and HDP-UTILS Base URLs are the same as the baseurl=values in the HDP.repo file downloaded in Upgrade the Stack: Step 1 For a local repository, use the local repository Base

URL that you configured for the HDP Stack. For links to download the HDP repository files for your version of the Stack, see HDP Repositories.

3. Using the Ambari Web Services view, start the ZooKeeper service.

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

```
su -l {HDFS_USER} -c "/usr/lib/hadoop/sbin/hadoop-daemon.sh start
 journalnode"
```

Important

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

5. Because the file system version has now changed you must start the NameNode manually. On the NameNode host:

```
 su -l {HDFS_USER} -c "export HADOOP_LIBEXEC_DIR=/usr/lib/hadoop/libexec &&
 /usr/lib/hadoop/sbin/hadoop-daemon.sh start namenode -upgrade"
```

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

Note

In a NameNode HA configuration, this NameNode will not enter the standby state as usual. Rather, this NameNode will immediately enter the active state, perform an upgrade of its local storage directories, and also perform an upgrade of the shared edit log. At this point, the standby NameNode in the HA pair is still down. It will be out of sync with the upgraded active NameNode. To synchronize the active and standby NameNode, re-establishing HA, re-bootstrap the standbyNameNode by running the NameNode with the '-bootstrapStandby' flag. **Do NOT** start this standby NameNode with the '-upgrade' flag.

```
su -l {HDFS_USER} -c "hdfs namenode -bootstrapStandby -force"
```

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

6. Prepare the NameNode to work with Ambari:

   a. Open the Ambari Web GUI. If it has been open throughout the process, do a hard reset on your browser to force a reload.

   b. On the Services view, click **HDFS** to open the HDFS service.

c. Click **View Host** to open the NameNode host details page.

d. Use the dropdown menu to stop the NameNode.

e. On the Services view, restart the HDFS service. Make sure it passes the ServiceCheck. It is now under Ambari's control.

7. After the DataNodes are started, HDFS exits safemode. To monitor the status, run the following command:

```
sudo su -l $HDFS_USER -c "hdfs dfsadmin -safemode get"
```

Depending on the size of your system, a response may not display for up to 10 minutes. When HDFS exits safemode, the following message displays:

```
Safe mode is OFF
```

8. Make sure that the HDFS upgrade was successful. Execute steps 2 and 3 in Preparing for the Upgrade to create new versions of the logs and reports. Substitute "new" for "`old`" in the file names as necessary.

9. Compare the old and new versions of the following:

   - `dfs-old-fsck-1.log` versus `dfs-new-fsck-1.log`.

     The files should be identical unless the `hadoop fsck` reporting format has changed in the new version.

   - `dfs-old-lsr-1.log` versus `dfs-new-lsr-1.log`.

     The files should be identical unless the format of `hadoop fs -lsr` reporting or the data structures have changed in the new version.

   - `dfs-old-report-1.log` versus `fs-new-report-1.log`

     Make sure all DataNodes previously belonging to the cluster are up and running.

10. Update the configuration properties required for Application Timeline Server. Using Ambari Web -> Services -> Configs, choose a service, then add/modify values for each of the following properties:

```
YARN (Custom yarn-site.xml)
yarn.timeline-service.leveldb-timeline-store.path=/var/log/hadoop-yarn/
timeline
yarn.timeline-service.leveldb-timeline-store.ttl-interval-ms=300000
yarn.timeline-service.store-class=org.apache.hadoop.yarn.server.
applicationhistoryservice.timeline.LeveldbTimelineStore
yarn.timeline-service.ttl-enable=true
yarn.timeline-service.ttl-ms=2678400000
yarn.timeline-service.generic-application-history.store-class=org.apache.
hadoop.yarn.server.applicationhistoryservice.NullApplicationHistoryStore
yarn.timeline-service.webapp.address=
{PUT_THE_FQDN_OF_RESOURCEMANAGER_HOST_NAME_HERE}:8188
yarn.timeline-service.webapp.https.address=
{PUT_THE_FQDN_OF_RESOURCEMANAGER_HOST_NAME_HERE}:8190
yarn.timeline-service.address=0.0.0.0:10200
```

```
HIVE (hive-site.xml)
hive.execution.engine=mr
hive.exec.failure.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.post.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.exec.pre.hooks=org.apache.hadoop.hive.ql.hooks.ATSHook
hive.tez.container.size={map-container-size}
        *If mapreduce.map.memory.mb > 2GB then set it equal to mapreduce.
map.memory.*
        *Otherwise set it equal to mapreduce.reduce.memory.mb*
```

11. Use the Ambari Web Services view to start YARN.

12. Use the Ambari Web Services view to start MapReduce2.

13. Upgrade HBase:

    a. Make sure that all HBase components - RegionServers and HBase Master - are stopped.

    b. Use the Ambari Web Services view, start the ZooKeeper service. Wait until the ZK service is up and running.

    c. On the HBase Master host, make these configuration changes:

       i. In `HBASE_CONFDIR/hbase-site.xml`, set the property `dfs.client.read.shortcircuit` to `false`.

       ii. In the configuration file, find the value of the `hbase.tmp.dir` property and make sure that the directory exists and is readable and writeable for the HBase service user and group.

       ```
       chown -R $HBASE_USER:$HADOOP_GROUP $HBASE.TMP.DIR
       ```

       iii. Go to the Upgrade Folder and check in the saved global configuration file named `global_<$TAG>` for the value of the property `hbase_pid_dir` and `hbase_log_dir`. Make sure that the directories are readable and writeable for the HBase service user and group.

       ```
       chown -R $HBASE_USER:$HADOOP_GROUP $hbase_pid_dir
       chown -R $HBASE_USER:$HADOOP_GROUP $hbase_log_dir
       ```

       Do this on **every** host where a RegionServer is installed as well as on the HBase Master host.

       iv. Check for HFiles in V1 format. HBase 0.96.0 discontinues support for HFileV1. Before the actual upgrade, run the following command to check if there are HFiles in V1 format:

       ```
       hbase upgrade -check
       ```

       HFileV1 was a common format prior to HBase 0.94. You may see output similar to:

       ```
       Tables Processed:

       hdfs://localhost:41020/myHBase/.META.
       hdfs://localhost:41020/myHBase/usertable
       hdfs://localhost:41020/myHBase/TestTable
       hdfs://localhost:41020/myHBase/t
       ```

```
Count of HFileV1: 2
HFileV1:
hdfs://localhost:41020/myHBase/usertable/
fa02dac1f38d03577bd0f7e666f12812/family/249450144068442524
hdfs://localhost:41020/myHBase/usertable/
ecdd3eaee2d2fcf8184ac025555bb2af/family/249450144068442512

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

When you run the upgrade check, if "Count of HFileV1" returns any files, start the hbase shell to use major compaction for regions that have HFileV1 format. For example in the sample output above, you must compact the `fa02dac1f38d03577bd0f7e666f12812` and `ecdd3eaee2d2fcf8184ac025555bb2af` regions.

v. Upgrade HBase. You must be the HBase service user.

```
sudo su -l $HBASE_USER -c "hbase upgrade -execute"
```

Make sure that the output contains the string "Successfully completed Znode upgrade".

vi. Use the Services view to start the HBase service. Make sure that Service Check passes.

14. Upgrade Oozie.

Note

You must replace your Oozie configuration after upgrading.

a. Perform the following preparation steps on each oozie server and client:

i. Copy files from the backup folder conf to /etc/oozie/conf directory.

```
cp {temp.folder.name}/oozie-sit.xml /etc/oozie/conf
```

```
chmod -R 777 /etc/alternatives/oozie-tomcat-conf/conf
```

```
rm /usr/lib/oozie/conf
```

```
ln -s /etc/oozie/conf /usr/lib/oozie/conf
```

ii. Create `/usr/lib/oozie/libtext-{customer}` directory.

```
mkdir /usr/lib/oozie/libext-{customer}
```

iii. Grant read/write access to the Oozie user.

```
chmod -R 666 /usr/lib/oozie/libext-{customer}
```

iv. Copy the JDBC jar of your Oozie database to both `/usr/lib/oozie/libext-{customer}` and `/usr/lib/oozie/libtools`.

v. Copy these files to /usr/lib/oozie/libext-{customer} directory

```
cp /usr/lib/hadoop/lib/hadoop-lzo*.jar /usr/lib/oozie/libext-{customer}
```

```
cp /usr/share/HDP-oozie/ext-2.2.zip /usr/lib/oozie/libext-{customer}
```

b. Upgrade steps:

i. On the Services view, make sure YARN and MapReduce2 are running.

ii. Make sure that the Oozie service is stopped.

iii. Upgrade Oozie. You must be the Oozie service user. On the Oozie host:

```
sudo su -l $OOZIE_USER -c"/usr/lib/oozie/bin/ooziedb.sh upgrade -run"
```

Make sure that the output contains the string "Oozie DB has been upgrade to Oozie version *'OOZIE Build Version*'".

iv. Prepare the Oozie WAR file, run as root:

Note

The Oozie server must be **not** running for this step. If you get the message "ERROR: Stop Oozie first", it means the script still thinks it's running. Check, and if needed, remove the process id (pid) file indicated in the output.

```
sudo su -l oozie -c "/usr/lib/oozie/bin/oozie-setup.sh prepare-war -d /
usr/lib/oozie/libext-{customer}"
```

Make sure that the output contains the string "New Oozie WAR file with added".

v. Using Ambari Web UI **Services** > **Oozie** > **Configs**, expand **Advanced**, then edit the following properties:

A. In `oozie.service.coord.push.check.requeue.interval`, ***replace*** the existing property value with the following one:

```
30000
```

B. In `oozie.service.SchemaService.wf.ext.schemas`, ***append*** (using copy/paste) to the existing property value the following string:

```
shell-action-0.2.xsd,oozie-sla-0.1.xsd,oozie-sla-0.2.xsd,hive-
action-0.3.xsd
```

Note

If you have customized schemas, append this string to your custom schema name string.

Do not overwrite custom schemas.

If you have no customized schemas, you can replace the existing string with the following one:

shell-action-0.1.xsd,email-action-0.1.xsd,hive-action-0.2.xsd, sqoop-action-0.2.xsd,ssh-action-0.1.xsd,distcp-action-0.1.xsd,shell-action-0.2.xsd,oozie-sla-0.1.xsd,oozie-sla-0.2.xsd,hive-action-0.3.xsd

C. In `oozie.service.URIHandlerService.uri.handlers`, ***append*** to the existing property value the following string:

```
org.apache.oozie.dependency.FSURIHandler,org.apache.oozie.dependency.
HCatURIHandler
```

D. In `oozie.services`, **append** to the existing property value the following string:

```
org.apache.oozie.service.XLogStreamingService,org.apache.oozie.
service.JobsConcurrencyService
```

Note

If you have customized properties, append this string to your custom property value string.

Do not overwrite custom properties.

If you have no customized properties, you can replace the existing string with the following one:

org.apache.oozie.service.SchedulerService,
org.apache.oozie.service.InstrumentationService,
org.apache.oozie.service.CallableQueueService,
org.apache.oozie.service.UUIDService,
org.apache.oozie.service.ELService,
org.apache.oozie.service.AuthorizationService,
org.apache.oozie.service.UserGroupInformationService,
org.apache.oozie.service.HadoopAccessorService,
org.apache.oozie.service.URIHandlerService,
org.apache.oozie.service.MemoryLocksService,
org.apache.oozie.service.DagXLogInfoService,
org.apache.oozie.service.SchemaService,
org.apache.oozie.service.LiteWorkflowAppService,
org.apache.oozie.service.JPAService,
org.apache.oozie.service.StoreService,
org.apache.oozie.service.CoordinatorStoreService,

org.apache.oozie.service.SLAStoreService,
org.apache.oozie.service.DBLiteWorkflowStoreService,
org.apache.oozie.service.CallbackService,
org.apache.oozie.service.ActionService,
org.apache.oozie.service.ActionCheckerService,
org.apache.oozie.service.RecoveryService,
org.apache.oozie.service.PurgeService,
org.apache.oozie.service.CoordinatorEngineService,
org.apache.oozie.service.BundleEngineService,
org.apache.oozie.service.DagEngineService,
org.apache.oozie.service.CoordMaterializeTriggerService,
org.apache.oozie.service.StatusTransitService,
org.apache.oozie.service.PauseTransitService,
org.apache.oozie.service.GroupsService,
org.apache.oozie.service.ProxyUserService,
org.apache.oozie.service.XLogStreamingService,
org.apache.oozie.service.JobsConcurrencyService

E. In `oozie.services.ext`, ***append*** to the existing property value the following string:

```
org.apache.oozie.service.PartitionDependencyManagerService,org.
apache.oozie.service.HCatAccessorService
```

F. After modifying all properties on the Oozie Configs page, scroll down, then choose **Save** to update oozie.site.xml, using the modified configurations.

vi. Replace the content of `/usr/oozie/share` in HDFS. On the Oozie server host:

A. Extract the Oozie sharelib into a `tmp` folder.

```
mkdir -p /tmp/oozie_tmp
cp /usr/lib/oozie/oozie-sharelib.tar.gz /tmp/oozie_tmp
cd /tmp/oozie_tmp
tar xzvf oozie-sharelib.tar.gz
```

B. Back up the`/user/oozie/share` folder in HDFS and then delete it. If you have any custom files in this folder back them up separately and then add them back after the share folder is updated.

```
su -l $HDFS_USR -c "$hdfs dfs -copyToLocal /user/oozie/share /tmp/
oozie_tmp/oozie_share_backup"
su -l $HDFS_USR -c "$hdfs dfs -rm -r /user/oozie/share"
```

C. Add the latest share libs that you extracted in step 1. After you have added the files, modify ownership and acl.

```
su -l $HDFS_USR -c "hdfs dfs -copyFromLocal /tmp/oozie_tmp/share /
user/oozie/."
su -l $HDFS_USR -c "hdfs dfs -chown -R oozie:hadoop /user/oozie"
su -l $HDFS_USR -c "hdfs dfs -chmod -R 755 /user/oozie"
```

vii.Use the Services view to start the Oozie service. Make sure that ServiceCheck passes for Oozie.

15.Update WebHcat.

a. Modify the `webhcat-site` config type.

   On the Ambari server, use `/var/lib/ambari-server/resources/scripts/configs.sh` to modify configuration properties in templeton.storage.class:

```
configs.sh set $HOSTNAME $CLUSTERNAME $CONFIGURATION-TYPE $PROPERTY-NAME
 $PROPERTY-VALUE
For example: configs.sh set <yourhostname> <yourclustername> webhcat-
site "templeton.storage.class" "org.apache.hive.hcatalog.templeton.tool.
ZooKeeperStorage"
```

b. Update the Pig and Hive tar bundles, by updating the following files:

   - /apps/webhcat/pig.tar.gz
   - /apps/webhcat/hive.tar.gz

   Note

   You will find these files on a host where webhcat is installed.

   For example, to update a *.tar.gz file:

   i. Move the file to a local directory.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -copyToLocal /
apps/webhcat/*.tar.gz ${local_backup_dir}"
```

   ii. Remove the old file.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -rm /apps/
webhcat/*.tar.gz"
```

   iii. Copy the new file.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -
copyFromLocal /usr/share/HDP-webhcat/*.tar.gz /apps/webhcat"
```

c. Update /app/webhcat/hadoop-streaming.jar file.

   i. Move the file to a local directory.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -copyToLocal /
apps/webhcat/hadoop-streaming*.jar ${local_backup_dir}"
```

   ii. Remove the old file.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -rm /apps/
webhcat/hadoop-streaming*.jar"
```

   iii. Copy the new hadoop-streaming.jar file.

```
su -l $HCAT_USR -c "hadoop --config /etc/hadoop/conf fs -
copyFromLocal /usr/lib/hadoop-mapreduce/hadoop-streaming*.jar /apps/
webhcat"
```

16. Make sure Ganglia no longer attempts to monitor JobTracker.

a. Make sure Ganglia is stopped.

b. Log into the host where JobTracker was installed (and where ResourceManager is installed after the upgrade).

c. Backup the folder `/etc/ganglia/hdp/HDPJobTracker` .

d. Remove the folder `/etc/ganglia/hdp/HDPJobTracker`.

e. Remove the folder `$ganglia_runtime_dir/HDPJobTracker`.

Note

For the value of `$ganglia_runtime_dir`, in the Upgrade Folder, check the saved global configuration file `global_<$TAG>`.

17. Use the Services view to start the remaining services back up.

18. The upgrade is now fully functional but not yet finalized. Using the `finalize` command removes the previous version of the NameNode and DataNode storage directories.

Important

After the upgrade is finalized, the system cannot be rolled back. Usually this step is not taken until a thorough testing of the upgrade has been performed.

The upgrade must be finalized before another upgrade can be performed.

Note

Directories used by Hadoop 1 services set in /etc/hadoop/conf/taskcontroller.cfg are not automatically deleted after upgrade. Administrators can choose to delete these directories after the upgrade.

To finalize the upgrade:

```
sudo su -l $HDFS_USER -c "hadoop dfsadmin -finalizeUpgrade"
```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

4. Upgrading the HDP Stack to 1.3.3

The stack is the coordinated set of Hadoop components that you have installed. If you have a current instance of the 1.2.0/1.2.1 stack that was installed and managed by Ambari that you want to upgrade to the current 1.3.3 version of the stack and to also upgrade to the 1.2.5 version of Ambari Server and Agents, use the following instructions. This insures that the upgraded stack can still be managed by Ambari.

If you are upgrading from the 1.3.0 stack to the 1.3.3 stack, use Section 5: Upgrading the Stack (from 1.3.0 to 1.3.3), not Section 4: Upgrading the Stack (from 1.2.* to 1.3.3).

Note

If you have already upgraded to Ambari Server 1.2.5 and just want to upgrade the HDP stack, you can skip Sections 4.2 and 4.3.

4.1. Preparing for the Upgrade

Use the following steps to prepare your system for the upgrade.

1. If you are upgrading Ambari as well as the stack, you must know the location of the Nagios and Ganglia servers for that process. Use the **Services**->**Nagios/Ganglia**-> **Summary** panel to locate the hosts on which they are running.

2. Use the **Services** view on the **Ambari Web** UI to stop all services, including MapReduce and all clients, running on HDFS. Do **not** stop HDFS yet.

3. Create the following logs and other files.

   Because the upgrade to 1.3.3 stack includes a version upgrade of HDFS, creating these logs allows you to check the integrity of the file system post upgrade. While this is not absolutely necessary, doing so is strongly encouraged.

   a. Run `fsck` with the following flags and send the results to a log. The resulting file contains a complete block map of the file system. You use this log later to confirm the upgrade.

   ```
   su $HDFS_USER
   hadoop fsck / -files -blocks -locations > /tmp/dfs-old-fsck-1.log
   ```

   where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

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

   ```
   su $HDFS_USER
   hadoop dfs -lsr / > /tmp/dfs-old-lsr-1.log
   ```

   where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

   c. Create a list of all the DataNodes in the cluster.

   ```
   su $HDFS_USER
   hadoop dfsadmin -report > /tmp/dfs-old-report-1.log
   ```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

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

e. Optional: create the logs again and check to make sure the results are identical.

4. Save the namespace. You must be the HDFS service user to do this and you must put the cluster in Safe Mode.

   ```
   hadoop dfsadmin -safemode enter
   hadoop dfsadmin -saveNamespace
   ```

5. Copy the following checkpoint files into a backup directory. You can find the directory by using the **Services View** in the UI. Select the **HDFS** service, the **Configs** tab, in the Namenode section, look up the property **NameNode Directories**. It will be on your NameNode host.

   - `dfs.name.dir/edits` // depending on your system, may not exist
   - `dfs.name.dir/image/fsimage`

6. Stop HDFS. Make sure all services in the cluster are completely stopped.

7. If you are upgrading Hive, back up the Hive database.

8. Stop Ambari Server. On the Server host:

   ```
   ambari-server stop
   ```

9. Stop Ambari Agents. On each host:

   ```
   ambari-agent stop
   ```

4.2. Setting Up the Ambari Repository

This process prepares the updated repository.

1. Check to see if you have a `conf.save` directory for Ambari server and agents. If you do, move them to a back-up location:

   ```
   mv /etc/ambari-server/conf.save/ /etc/ambari-server/conf.save.bak
   ```

   ```
   mv /etc/ambari-agent/conf.save/ /etc/ambari-agent/conf.save.bak
   ```

2. Get the new Ambari bits. Use `wget` to fetch the repository file and replace the old repo file with the new repo file on every host.

   Important

   Check your current directory before you download the new repo file to make sure that there are no previous versions of the file. If you do not, and a previous version exists, the new download is saved with a numeric extension such as `ambari.repo.1`. Make sure that the version you copy is the new version.

- For RHEL/CentOS/Oracle Linux 5

```
wget http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/1.5.
1/ambari.repo
cp ambari.repo /etc/yum.repos.d/ambari.repo
```

- For RHEL/CentOS/Oracle Linux 6

```
wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.5.
1/ambari.repo
cp ambari.repo /etc/yum.repos.d/ambari.repo
```

- For SLES 11

```
wget http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.5.1/
ambari.repo
cp ambari.repo /etc/zypp/repos.d/ambari.repo
```

Note

If your cluster does not have access to the Internet, you need to set up a local repository with this data before you continue. See Configure the Local Repositories for more information.

4.3. Upgrading Ambari

This process upgrades Ambari Server, Ambari Agents, Ganglia, and Nagios.

1. Upgrade Ambari Server. From the Ambari Server host:

   - RHEL/CentOS/Oracle Linux

   ```
   yum clean all
   yum upgrade ambari-server ambari-log4j
   ```

   - SLES

   ```
   zypper clean
   zypper up ambari-server ambari-log4j
   ```

2. Check for upgrade success:

   - As the process runs, the console should produce output similar, although not identical, to this:

   ```
   Setting up Upgrade Process
   Resolving Dependencies
   --> Running transaction check
   ---> Package ambari-server.x86_64 0:1.2.2.3-1 will be updated
   ---> Package ambari-server.x86_64 0:1.2.2.4-1 will be updated ...
   ---> Package ambari-server.x86_64 0:1.2.2.5-1 will be an update ...
   ```

   After the process is complete, check each host to make sure the new 1.2.5 files have been installed:

   ```
   rpm -qa | grep ambari
   ```

- If the upgrade fails, the console should produce output similar to this:

  ```
  Setting up Upgrade Process
  No Packages marked for Update
  ```

3. Check to see if you have a folder named `/etc/ambari-server/conf.save`. If you do, rename it back:

   ```
   mv /etc/ambari-server/conf.save /etc/ambari-server/conf
   ```

4. Upgrade the Ambari Server schema. From the Ambari Server host:

   ```
   ambari-server upgrade
   ```

5. Upgrade the Ambari Agent on all hosts. From each Ambari Agent host:

   - RHEL/CentOS/Oracle Linux

     ```
     yum upgrade ambari-agent ambari-log4j
     ```

   - SLES

     ```
     zypper up ambari-agent ambari-log4j
     ```

   Note

   If you get a warning that begins "There are some running programs that use files deleted by recent upgrade," you can ignore it.

6. Check to see if you have a folder named `/etc/ambari-agent/conf.save` on each Agent host. If you do, rename it back. On each Agent host:

   ```
   mv /etc/ambari-agent/conf.save /etc/ambari-agent/conf
   ```

7. Upgrade Ganglia and Nagios:

   - Upgrade Ganglia Server. From the Ganglia Server host:

     - RHEL/CentOS/Oracle Linux

       ```
       yum upgrade ganglia-gmond ganglia-gmetad libganglia
       yum erase gweb hdp_mon_ganglia_addons
       yum install ganglia-web
       ```

     - SLES

       ```
       zypper up ganglia-gmond ganglia-gmetad libganglia
       zypper remove gweb hdp_mon_ganglia_addons
       zypper install ganglia-web
       ```

   - Upgrade Ganglia Monitor. From every host that has Ganglia Monitor installed:

     - RHEL/CentOS/Oracle Linux

       ```
       yum upgrade ganglia-gmond libganglia
       ```

     - SLES

```
zypper up ganglia-gmond libganglia
```

- Upgrade Nagios. From the Nagios Server host:
  - RHEL/CentOS/Oracle Linux

    ```
    yum upgrade nagios
    yum upgrade hdp_mon_nagios_addons
    yum erase nagios-plugins-1.4.15
    yum install nagios-plugins-1.4.9
    ```

    The 1.4.9 version of the plugin may already be installed. In this case, the second step is a no-op.
  - SLES

    ```
    zypper up nagios
    zypper up hdp_mon_nagios_addons
    zypper remove nagios-plugins-1.4.15
    zypper install nagios-plugins-1.4.9
    ```

    The 1.4.9 version of the plugin may already be installed. In this case, the second step is a no-op.

4.4. Upgrading the Stack (from 1.2.* to 1.3.3)

1. Update the stack version in the Server database, depending on if you are using a local repository:

   ```
   ambari-server upgradestack HDP-1.3.3
   ```

2. Upgrade the HDP repository on all hosts and replace the old repo file with the new file:

   Important

   The file you download is named `hdp.repo`. To function properly in the system, it must be named `HDP.repo`. Once you have completed the "mv" of the new repo file to the repos.d folder, make sure there is no file named `hdp.repo` anywhere in your repos.d folder.

   - For RHEL/CentOS/Oracle Linux 5

     ```
     wget  http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/1.3.3.
     0/hdp.repo
     mv hdp.repo /etc/yum.repos.d/HDP.repo
     ```

   - For RHEL/CentOS/Oracle Linux 6

     ```
     wget http://public-repo-1.hortonworks.com/HDP/centos6/1.x/updates/1.3.3.0/
     hdp.repo
     mv hdp.repo /etc/yum.repos.d/HDP.repo
     ```

   - For SLES 11

     ```
     wget  http://public-repo-1.hortonworks.com/HDP/suse11/1.x/updates/1.3.3.0/
     hdp.repo
     ```

```
mv hdp.repo /etc/zypp/repos.d/HDP.repo
```

3. Upgrade the stack on all Agent hosts. Skip any components your installation does not use:

   • For RHEL/CentOS/Oracle Linux

   a. Upgrade the following components:

   ```
   yum upgrade "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
    "sqoop*" "zookeeper*" "hbase*" "hive*" "hcatalog*" "webhcat-tar*"
   hdp_mon_nagios_addons
   ```

   b. Check to see that the components in that list are upgraded.

   ```
   yum list installed | grep HDP-$old-stack-version-number
   ```

   None of the components from that list should appear in the returned list.

   c. Upgrade Oozie, if you are using Oozie:

   ```
   rpm -e --nopostun oozie-$old_version_number
   yum install oozie
   ```

   You can get the value of $old_version_number from the output of the previous step.

   d. Upgrade Oozie Client:

   ```
   yum upgrade oozie-client
   ```

   e. Upgrade ZooKeeper.

   i. Check to see if ZooKeeper needs upgrading.

   ```
   yum list installed | grep zookeeper
   ```

   If the displayed version number is **not** 3.4.5.1.3.2.0, you need to upgrade.

   ii. Because HBase depends on ZooKeeper, deleting the current version of ZooKeeper automatically deletes the current version of HBase. It must be re-installed. Check to see if HBase is currently installed.

   ```
   yum list installed | grep hbase
   ```

   iii. Delete the current version of ZooKeeper.

   ```
   yum erase zookeeper
   ```

   iv. Install ZooKeeper.

   ```
   yum install zookeeper
   ```

   v. If you need to, re-install HBase.

   ```
   yum install hbase
   ```

   vi. Check to see if all components have been upgraded.

```
yum list installed | grep HDP-$old-stack-version-number
```

The only non-upgraded component you may see in this list is `extjs`, which does not need to be upgraded.

- For SLES

  a. Upgrade the following components:

  ```
  zypper up collectd epel-release* gccxml* pig* hadoop* sqoop* hive*
   hcatalog* webhcat-tar* hdp_mon_nagios_addons*
  yast --update hadoop hcatalog hive
  ```

  b. Upgrade ZooKeeper and HBase.

  ```
  zypper update zookeeper-3.4.5.1.3.2.0
  zypper remove zookeeper
  zypper se -s zookeeper
  ```

  You should see ZooKeeper v3.4.5.1.3.2.0 in the output.

  Install ZooKeeper v3.4.5.1.3.2.0:

  ```
  zypper install zookeeper-3.4.5.1.3.2.0
  ```

  This command also uninstalls HBase. Now use the following commands to install HBase:

  ```
  zypper install hbase-0.94.6.1.3.2.0
  zypper update hbase
  ```

  c. Upgrade Oozie:

  ```
  rpm -e --nopostun oozie-$old_version_number
  zypper update oozie-3.3.2.1.3.2.0
  zypper remove oozie
  zypper se -s oozie
  ```

  You should see Oozie v3.3.2.1.3.2.0 in the output.

  Install Oozie v3.3.2.1.3.2.0:

  ```
  zypper install oozie-3.3.2.1.3.2.0
  ```

4. Start the Ambari Server. On the Server host:

   ```
   ambari-server start
   ```

5. Start each Ambari Agent. On all Agent hosts:

   ```
   ambari-agent start
   ```

6. Because the file system version has now changed you must start the NameNode manually. On the NameNode host:

   ```
   sudo su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start
    namenode -upgrade"
   ```

   Depending on the size of your system, this step may take up to 10 minutes.

7. Track the status of the upgrade:

   ```
   hadoop dfsadmin -upgradeProgress status
   ```

   Continue tracking until you see:

   ```
   Upgrade for version -44 has been completed.
   Upgrade is not finalized.
   ```

   Note

   You finalize the upgrade later. DO NOT run the balancer before finalizing an upgrade. No block deletion occurs until you finalize the upgrade. Running the balancer before finalizing an upgrade may duplicate data blocks, and increase disk usage.

8. Open the Ambari Web GUI. If you have continued to run the Ambari Web GUI, do a hard reset on your browser. Use **Services View** to start the HDFS service. This starts the SecondaryNameNode and the DataNodes.

9. After the DataNodes are started, HDFS exits safemode. To monitor the status:

   ```
   hadoop dfsadmin -safemode get
   ```

   Depending on the size of your system, this may take up to 10 minutes or so. When HDFS exits safemode, this is displayed as a response to the command:

   ```
   Safe mode is OFF
   ```

10. Make sure that the HDFS upgrade was successful. Go through steps 2 and 3 in Preparing for the Upgrade to create new versions of the logs and reports. Substitute "`new`" for "`old`" in the file names as necessary

11. Compare the old and new versions of the following:

    - `dfs-old-fsck-1.log` versus `dfs-new-fsck-1.log`.

      The files should be identical unless the `hadoop fsck` reporting format has changed in the new version.

    - `dfs-old-lsr-1.log` versus `dfs-new-lsr-1.log`.

      The files should be identical unless the the format of `hadoop fs -lsr` reporting or the data structures have changed in the new version.

    - `dfs-old-report-1.log` versus `fs-new-report-1.log`

      Make sure all DataNodes previously belonging to the cluster are up and running.

12. Use the Ambari Web **Services** view-> Services Navigation->**Start All** to start services back up.

13. The upgrade is now fully functional but not yet finalized. Using the `finalize` command removes the previous version of the NameNode and DataNode's storage directories.

Important

After the upgrade is finalized, the system cannot be rolled back. Usually this step is not taken until a thorough testing of the upgrade has been performed.

The upgrade must be finalized before another upgrade can be performed.

Note

Directories used by Hadoop 1 services set in /etc/hadoop/conf/taskcontroller.cfg are not automatically deleted after upgrade. Administrators can choose to delete these directories after the upgrade.

To finalize the upgrade:

```
sudo su -l $HDFS_USER -c "hadoop dfsadmin -finalizeUpgrade"
```

where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

4.5. Upgrading the Stack (from 1.3.0 to 1.3.3)

1. Update the stack version in the Server database, depending on if you are using a local repository:

   ```
   ambari-server upgradestack HDP-1.3.3
   ```

2. Upgrade the HDP repository on all hosts and replace the old repo file with the new file:

   ### Important

   The file you download is named `hdp.repo`. To function properly in the system, it must be named `HDP.repo`. After you have completed the "mv" of the new repo file to the repos.d folder, make sure there is no file named `hdp.repo` anywhere in your repos.d folder.

   - For RHEL/CentOS/Oracle Linux 5

     ```
     wget  http://public-repo-1.hortonworks.com/HDP/centos5/1.x/updates/1.3.3.
     0/hdp.repo
     mv hdp.repo /etc/yum.repos.d/HDP.repo
     ```

   - For RHEL/CentOS/Oracle Linux 6

     ```
     wget http://public-repo-1.hortonworks.com/HDP/centos6/1.x/updates/1.3.3.0/
     hdp.repo
     mv hdp.repo /etc/yum.repos.d/HDP.repo
     ```

   - For SLES 11

     ```
     wget  http://public-repo-1.hortonworks.com/HDP/suse11/1.x/updates/1.3.3.0/
     hdp.repo

     mv hdp.repo /etc/zypp/repos.d/HDP.repo
     ```

3. Upgrade the stack on all Agent hosts. Skip any components your installation does not use:

   - For RHEL/CentOS/Oracle Linux

     a. Upgrade the following components:

        ```
        yum upgrade "collectd*" "epel-release*" "gccxml*" "pig*" "hadoop*"
         "sqoop*" "zookeeper*" "hbase*" "hive*" "hcatalog*" "webhcat-tar*"
        "oozie*" hdp_mon_nagios_addons
        ```

     b. Check to see if those components have been upgraded:

        ```
        yum list installed | grep HDP-$old-stack-version-number
        ```

        The only non-upgraded component you may see in this list is `extjs`, which does not need to be upgraded.

   - For SLES

     a. Upgrade the following components:

        ```
        zypper up collectd gccxml* pig* hadoop* sqoop* hive* hcatalog* webhcat-
        tar* zookeeper* oozie* hbase* hdp_mon_nagios_addons*
        yast --update hadoop hcatalog hive
        ```

4. Start the Ambari Server. On the Server host:

   ```
   ambari-server start
   ```

5. Start each Ambari Agent. On all Agent hosts:

   ```
   ambari-agent start
   ```

6. Because the file system version has now changed you must start the NameNode manually. On the NameNode host:

   ```
   sudo su -l $HDFS_USER -c "/usr/lib/hadoop/bin/hadoop-daemon.sh start
    namenode -upgrade"
   ```

   Depending on the size of your system, this step may take up to 10 minutes.

7. Track the status of the upgrade:

   ```
   hadoop dfsadmin -upgradeProgress status
   ```

   Continue tracking until you see:

   ```
   Upgrade for version -44 has been completed.
   Upgrade is not finalized.
   ```

   ### Note

   You finalize the upgrade later.

8. Open the Ambari Web GUI. If you have continued to run the Ambari Web GUI, do a hard reset on your browser. Use **Services View** to start the HDFS service. This starts the SecondaryNameNode and the DataNodes.

9. After the DataNodes are started, HDFS exits safemode. To monitor the status:

   ```
   hadoop dfsadmin -safemode get
   ```

   Depending on the size of your system, this may take up to 10 minutes or so. When HDFS exits safemode, this is displayed as a response to the command:

   ```
   Safe mode is OFF
   ```

10. Make sure that the HDFS upgrade succeeded. Go through steps 2 and 3 in Preparing for the Upgrade to create new versions of the logs and reports. Substitute "`new`" for "`old`" in the file names as necessary

11. Compare the old and new versions of the following files:

    - `dfs-old-fsck-1.log` versus `dfs-new-fsck-1.log`.

      The files should be identical unless the `hadoop fsck` reporting format has changed in the new version.

    - `dfs-old-lsr-1.log` versus `dfs-new-lsr-1.log`.

      The files should be identical unless the the format of `hadoop fs -lsr` reporting or the data structures have changed in the new version.

    - `dfs-old-report-1.log` versus `fs-new-report-1.log`

      Make sure all DataNodes previously belonging to the cluster are up and running.

12. Use the Ambari Web **Services** view-> Services Navigation->**Start All** to start services back up.

13. The upgrade is now fully functional but not yet finalized. Using the `finalize` comand removes the previous version of the NameNode and DataNode storage directories.

    Important

    After the upgrade is finalized, the system cannot be rolled back. Do not take this step until you perform thorough testing of the upgrade.

    The upgrade must be finalized, however, before another upgrade can be performed.

    Note

    Directories used by Hadoop 1 services set in /etc/hadoop/conf/taskcontroller.cfg are not automatically deleted after upgrade. Administrators can choose to delete these directories after the upgrade.

    To finalize the upgrade:

    ```
    sudo su -l $HDFS_USER -c "hadoop dfsadmin -finalizeUpgrade"
    ```

    where *$HDFS_USER* is the HDFS Service user (by default, `hdfs`).

5. Upgrading Operating Systems on an Ambari-based Hadoop Installation

Ambari requires specific versions of the files for components that it uses. There are three steps you should take to make sure that these versions continue to be available:

- Disable automatic OS updates
- Do not update any HDP components such as MySQL, Ganglia, etc.
- If you must perform an OS update, do a manual kernel update only.

6. Upgrading From Older Ambari Server versions to 1.2.5

This process upgrades Ambari Server from older, 1.x versions to version 1.2.5. It does not change the underlying Hadoop Stack. This is a 12-step manual process.

Note

You must know the location of the Nagios server for Step 9. Use the **Services View**-> **Summary** panel to locate the host on which it is running.

1. Stop the Ambari Server and all Ambari Agents. From the Ambari Server host:

   ```
   ambari-server stop
   ```

   From each Ambari Agent host:

   ```
   ambari-agent stop
   ```

2. Get the new Ambari bits. Use `wget` to fetch the repository file and replace the old repo file with the new repo file on every host.

   - Fetch the new repo file:

     For RHEL/CentOS 5/Oracle Linux 5

     ```
     wget http://public-repo-1.hortonworks.com/ambari/centos5/1.x/updates/1.2.
     5.17/ambari.repo
     ```

     For RHEL/CentOS 6/Oracle Linux 6

     ```
     wget http://public-repo-1.hortonworks.com/ambari/centos6/1.x/updates/1.2.
     5.17/ambari.repo
     ```

     For SLES 11

     ```
     wget http://public-repo-1.hortonworks.com/ambari/suse11/1.x/updates/1.2.5.
     17/ambari.repo
     ```

     ### Important

     Check your current directory before you download the new repo file to make sure that there are no previous versions of the file. If you do not, and a previous version exists, the new download will be saved with a numeric extension such as `ambari.repo.1`. Make sure that the version you copy is the new version.

   - Replace the old repo file with the new repo file.

     For RHEL/CentOS 5/Oracle Linux 5

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

     For RHEL/CentOS 6/Oracle Linux 6

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

For SLES 11

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

Note

If your cluster does not have access to the Internet, set up a local repository with this data before you continue. See Configure the Local Repositories for more information.

3. Upgrade Ambari Server. From the Ambari Server host:

   - RHEL/CentOS/Oracle Linux

     ```
     yum clean all
     yum upgrade ambari-server-1.2.5.17 ambari-log4j-1.2.5.17
     ```

   - SLES

     ```
     zypper clean
     zypper up ambari-server-1.2.5.17 ambari-log4j-1.2.5.17
     ```

4. Check for upgrade success:

   - As the process runs, the console should produce output similar, although not identical, to this:

     ```
     Setting up Upgrade Process
     Resolving Dependencies
     --> Running transaction check
     ---> Package ambari-server.x86_64 0:1.2.2.3-1 will be updated
     ---> Package ambari-server.x86_64 0:1.2.2.4-1 will be updated ...
     ---> Package ambari-server.x86_64 0:1.2.2.5-1 will be an update ...
     ```

     After the process is complete, check each host to make sure the new 1.2.4 files have been installed:

     ```
     rpm -qa | grep ambari
     ```

   - If the upgrade fails, the console should produce output similar to this:

     ```
     Setting up Upgrade Process
     No Packages marked for Update
     ```

5. Check to see if you have a folder named `/etc/ambari-server/conf.save`. If you do, rename it back:

   ```
   mv /etc/ambari-server/conf.save /etc/ambari-server/conf
   ```

6. Upgrade the Ambari Server schema. From the Ambari Server host:

   ```
   ambari-server upgrade
   ```

7. Upgrade the Ambari Agent on all hosts. From each Ambari Agent host:

   - RHEL/CentOS/Oracle Linux

```
yum upgrade ambari-agent ambari-log4j
```

- SLES

```
zypper up ambari-agent ambari-log4j
```

Note

If you get a warning that begins "There are some running programs that use files deleted by recent upgrade" you can ignore it.

8. Check to see if you have a file named `/etc/ambari-agent/conf.save` on each Agent host. If you do, rename it back. On each Agent host:

```
mv /etc/ambari-agent/conf.save /etc/ambari-agent/conf
```

9. Upgrade the Nagios and Ganglia addons package and restart. On the Nagios/Ganglia host:

   - RHEL/CentOS/Oracle Linux

   ```
   yum upgrade hdp_mon_nagios_addons hdp_mon_ganglia_addons
   service httpd restart
   ```

   - SLES

   ```
   zypper up hdp_mon_nagios_addons hdp_mon_ganglia_addons
   service apache2 restart
   ```

10. Start the Server and the Agents on all hosts. From the Server host:

```
ambari-server start
```

From each Agent host:

```
ambari-agent start
```

11. Open **Ambari Web**. Point your browser to `http://{your.ambari.server}:8080`

Important

You need to refresh your browser so that it loads the new version of the code. Hold the Shift key down while clicking the refresh button on the browser. If you have problems, clear your browser cache manually and restart Ambari Server.

Use the Admin name and password you have set up to log in.

12. Re-start the Ganglia, Nagios, and MapReduce services. In **Ambari Web**.

   a. Go to the **Services View** and select each service.

   b. Use the **Management Header** to stop and re-start each service.

7. Fixing a Failed Ambari Server Upgrade

If upgrading Ambari Server fails, use the instructions in one of the following sections to fix the failed upgrade.

- Upgrade Failure with PostreSQL
- Upgrade Failure with Oracle
- Upgrade Failure from a Local Repository

7.1. Upgrade Failure with PostgreSQL

If you installed Ambari server with a PostgresQL database and upgrading Ambari Server using a remote or public repository failed, use the following steps to fix the upgrade.

1. Upgrade the database schema.

   ```
   var/lib/ambari-server/resources/upgrade/ddl/Ambari-DDL-Postgres-UPGRADE-1.3.
   0.sql

   with the following parameters:
   dbname = ambari
   username = ambari
   ```

2. Check database consistency.

   ```
   var/lib/ambari-server/resources/upgrade/ddl/Ambari-DDL-Postgres-UPGRADE-1.3.
   0.Check.sql

   with the following parameter:
   dbname = ambari
   ```

3. If you find an inconsistency, fix it using

   ```
   var/lib/ambari-server/resources/upgrade/ddl/Ambari-DDL-Postgres-UPGRADE-1.3.
   0.Fix.sql

   with the following parameter:
   dbname = ambari
   ```

7.2. Upgrade Failure with Oracle

If you installed Ambari server with an Oracle database and upgrading Ambari Server using a remote or public repository failed, run the following script to upgrade the database schema.

```
/var/lib/ambari-server/resources/upgrade/ddl/Ambari-DDL-Oracle-UPGRADE.sql
```

7.3. Upgrade Failure from a Local Repository

If you install and upgrade Ambari server using a local repository and upgrading Ambari Server using your local repository failed, find information necessary to fix the upgrade in the following locations:

1. Check for local repo version customized in `/var/lib/ambari-server/resources/stacks/HDPLocal`.
2. Check for the repo version used when creating the cluster, in `repos/repoinfo.xml`.
3. If local repo version is NOT the same as the NON-LOCAL repo version: Note the os, version, repoid and baseurl, found in `repos/repoinfo.xml`.

Then, choose the fix appropriate for your database version.

- PostgreSQL
- Oracle

7.3.1. Ambari Server with PostgreSQL

To fix a local Ambari/PostgreSQL upgrade:

1. Repair metadata information.

```
var/lib/ambari-server/resources/upgrade/dml/Ambari-DML-Postgres-
INSERT_METAINFO.sql

with the following parameters:

dbname = ambari
metainfo_key = repo:/HDP/<version>/<os>/<repoid>:baseurl
metainfo_value = <baseurl>
```

2. Run the fix script.

```
var/lib/ambari-server/resources/upgrade/dml/Ambari-DML-Postgres-
FIX_LOCAL_REPO.sql

with the following parameters:

dbname = ambari
```

7.3.2. Ambari Server with Oracle

To fix a local Ambari/PostgreSQL upgrade:

1. Repair metadata information.

```
var/lib/ambari-server/resources/upgrade/dml/Ambari-DML-Oracle-
INSERT_METAINFO.sql

with the following parameters:

argument #1 = repo:/HDP/<version>/<os>/<repoid>:baseurl
argument #2 = <baseurl>
```

2. Run the fix script.

```
var/lib/ambari-server/resources/upgrade/dml/Ambari-DML-Oracle-
FIX_LOCAL_REPO.sql
```