Hortonworks Data Platform

Apache Ambari Upgrade for IBM Power Systems

(October 30, 2017)

docs.hortonworks.com

Hortonworks Data Platform: Apache Ambari Upgrade for IBM Power Systems

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.

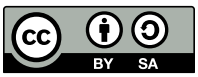


Table of Contents

List of Tables

1. Upgrading Ambari and HDP for IBM Power Systems

Ambari and the HDP Stack being managed by Ambari can be upgraded independently. It is always recommended to upgrade Ambari to the latest version before using it to upgrade HDP.

This guide provides information about upgrading Ambari to version 2.6.0.0 and then using Ambari to upgrade HDP to version 2.6.3.0 in an IBM Power Systems environment.:

2. Getting Ready to Upgrade Ambari and HDP

When preparing to upgrade Ambari and the HDP Cluster, we strongly recommend you review this checklist of items to confirm your cluster operation is healthy. Attempting to upgrade a cluster that is operating in an unhealthy state can produce unexpected results.

Important

Always upgrade Ambari to the latest version before upgrading the cluster.

- Ensure all services in the cluster are running.
- Run each Service Check (found under the Service Actions menu) and confirm they execute successfully.
- Clear all alerts, or understand why they are being generated. Remediate as necessary.
- Confirm start and stop for all services are executing successfully.
- Time service start and stops. The time to start and stop services is a big contributor to overall upgrade time so having this information handy is useful.
- Download the software packages prior to the upgrade. Place them in a local repository and/or consider using a storage proxy since multi-gigabyte downloads will be required on all nodes in the cluster.
- Ensure point-in-time backups are taken of all databases that support the cluster. This includes (among others) Ambari, Hive Metastore, Ranger and Oozie.

  For Ambari Upgrades

  - This (Ambari 2.6.x) Upgrade Guide will help you upgrade your existing Ambari server to version 2.6.x If you are upgrading to another Ambari version, please be sure to use the Ambari Upgrade Guide for that version.
  - Be sure to review the Known Issues and Behavioral Changes for this, Ambari-2.6.x release.

  For HDP Cluster Upgrades

  - Ensure sufficient disk space on `/usr/hdp/<version>` (roughly 3GB for each additional HDP release).
  - If you plan to add new services available with HDP to your cluster, the new services might include new service accounts. Any operational procedures required to support these new service accounts should be performed prior to the upgrade. The accounts will typically be required on all nodes in the cluster.
  - If your cluster includes Storm, document any running Storm topologies.

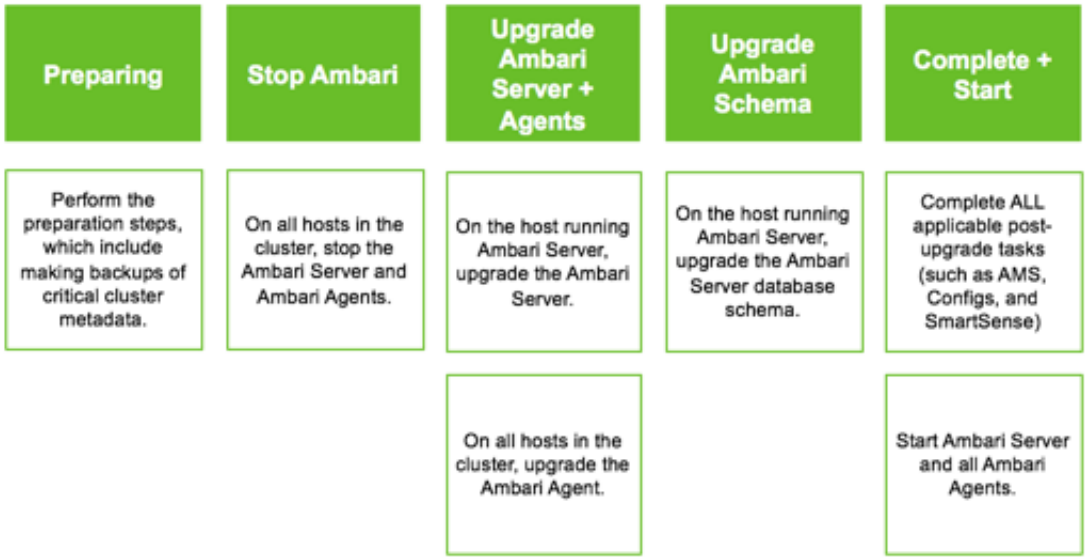
More Information

3. Upgrading Ambari

Ambari and the HDP cluster being managed by Ambari can be upgraded independently. This section describes the process to upgrade Ambari. You are **strongly encouraged** to read completely through this entire document before starting the upgrade process, to that you understand the interdependencies and order of the steps. It is **highly recommended** you validate these steps in a test environment to adjust + account for any special configurations for your cluster.

- Preparing to Upgrade [4]
- Upgrade Ambari [5]
- *Mandatory* Post-Upgrade Tasks [8]

The high-level process for upgrading Ambari is as follows:



Important

Completing post-upgrade tasks is mandatory.

3.1. Preparing to Upgrade

- Be sure to review the Ambari 2.6.0.0 release notes for Known Issues and Behavioral Changes.
- You **must** have root, administrative, or root-equivalent authorization on the Ambari server host and all servers in the cluster.
- You **must** backup the Ambari Server database.
- You **must** make a safe copy of the Ambari Server configuration file found at `/etc/ambari-server/conf/ambari.properties`.

- **Plan to upgrade the Ambari Metrics service:**
  - Record the location of the **Metrics Collector** component before you begin the upgrade process.
  - You **must** stop the Ambari Metrics service from **Ambari Web**.
  - After upgrading Ambari, you must also upgrade Ambari Metrics System and add the Grafana component.
- After upgrading Ambari, you must also upgrade SmartSense.

Note

During Ambari upgrade, the existing `/var/lib/ambari-server/ambari-env.sh` file is overwritten and a backup copy of `ambari-env.sh` (with extension `.rpmsave`) is created. If you have manually modified `ambari-env.sh` (for example, to change Ambari Server heap), you will need to re-apply your changes to the new file.

Next Steps

Upgrade Ambari [5]

More Information

Upgrade Ambari Metrics

Upgrade SmartSense

Ambari 2.6.0.0 Release Notes

Upgrade the Ambari Server version to 2.2

3.2. Upgrade Ambari

1. If you are running Ambari Metrics service in your cluster, stop the service. From **Ambari Web**, browse to **Services > Ambari Metrics** and select **Stop** from the **Service Actions** menu.
2. Stop the Ambari Server. On **the host** running Ambari Server:

```
ambari-server stop
```

3. Stop all Ambari Agents. On **each host** in your cluster running an Ambari Agent:

```
ambari-agent stop
```

4. Fetch the new Ambari repo and replace the old repository file with the new repository file **on all hosts** in your cluster.

Important

Check your current directory before you download the new repository file to make sure that there are no previous versions of the `ambari.repo` 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.

Select the repository appropriate for your environment:

```
wget -nv http://public-repo-1.hortonworks.com/ambari/centos7-ppc/2.x/
updates/2.6.0.0/ambari.repo -O /etc/yum.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.

Note

Ambari Server does not automatically turn off `iptables`. Check that your installation setup does not depend on `iptables` being disabled. After upgrading the server, you must either disable `iptables` manually or make sure that you have appropriate ports available on all cluster hosts.

5. Upgrade Ambari Server. On **the host** running Ambari Server:

   For RHEL/CentOS/Oracle Linux:

   ```
   yum clean all
   ```

   ```
   yum info ambari-server
   ```

   In the info output, visually validate that there is an available version containing "2.6"

   ```
   yum upgrade ambari-server
   ```

6. Check for upgrade success by noting progress during the Ambari Server installation process you started in Step 5.

   - As the process runs, the console displays output similar, although not identical, to the following:

     ```
     Setting up Upgrade Process Resolving Dependencies --> Running
     transaction check
     ```

   - If the upgrade fails, the console displays output similar to the following:

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

   - A successful upgrade displays output similar to the following:

     ```
     Updated: ambari-server.noarch 0:2.6.0-267 Complete!
     ```

   Note

   Confirm there is only one ambari-server*.jar file in `/usr/lib/ambari-server`. If there is more than one JAR file with name ambari-server*.jar, move all JARs except ambari-server-2.6.0.jar to `/tmp` before proceeding with upgrade.

7. Upgrade all Ambari Agents. On **each host** in your cluster running an Ambari Agent:

```
yum upgrade ambari-agent
```

8. After the upgrade process completes, check each host to make sure the new files have been installed:

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

9. Upgrade Ambari Server database schema. On **the host** running Ambari Server:

```
ambari-server upgrade
```

10. Start the Ambari Server. On **the host** running Ambari Server:

```
ambari-server start
```

11. Start all Ambari Agents. On **each host** in your cluster running an Ambari Agent:

```
ambari-agent start
```

12. Open Ambari Web.

    Point your browser to `http://<your.ambari.server>:8080`

    where <your.ambari.server> is the name of your ambari server host. For example, c6401.ambari.apache.org.

Important

Refresh your browser so that it loads the new version of the Ambari Web code. If you have problems, clear your browser cache manually, then restart Ambari Server.

13. Log in, using the Ambari administrator credentials that you have set up.

    For example, the default name/password is **admin/admin**.

    You will see a Restart indicator next to each service after upgrading. Ambari upgrade has added to/adjusted the configuration properties of your cluster based on new configuration types and properties being made available for each service with this release of Ambari. Review these changes by comparing the previous configuration with the latest version created by "ambari-upgrade".

14. If you have configured Ambari to authenticate against an external LDAP or Active Directory, you **must** re-run

```
ambari-server setup-ldap
```

15. If you have configured your cluster for Hive or Oozie with an external database (Oracle, MySQL or PostgreSQL), you **must** re-run

```
ambari-server setup --jdbc-db and --jdbc-driver
```

    to get the JDBC driver .jar file in place.

16. If you are running **Ambari Metrics** service in your cluster, you **must** upgrade Ambari Metrics System and add the Grafana component.

17. If your cluster includes the SmartSense service, you **must** upgrade SmartSense along with Ambari.

18. Perform any other post-upgrade tasks, as necessary.

Important

Completing post-upgrade tasks is mandatory.

Next Steps

Post-Upgrade Tasks ***Mandatory***

More Information

Using a Local Repository

Configuring Network Port Numbers

Set Up LDAP or Active Directory Authentication

Using Non-Default Databases - Hive

Using Non-Default Databases - Oozie

Upgrade Ambari Metrics

Upgrade SmartSense

3.3. *Mandatory* Post-Upgrade Tasks

Depending on the configuration of your cluster and your current Ambari version, you must upgrade any of the following features in your cluster, as described in the following topics:

Upgrading Ambari Infra	If your cluster includes Ambari Infra service, you must upgrade it along with Ambari.
Upgrading Ambari Log Search	If your cluster includes Ambari Log Search service, you must upgrade it along with Ambari.
Upgrading Ambari Metrics	If your cluster includes the Ambari Metrics System (AMS) service, you must upgrade the system along with Ambari. This will include adding the Grafana component to the system.
Upgrading Configurations	Certain scenarios may require that you modify configurations that Ambari did not upgrade automatically.

Upgrading SmartSense	If your cluster includes the SmartSense service, you must upgrade it after upgrading Ambari.

Next Steps

Restart services, only after you complete all applicable, post-upgrade tasks.

More Information

Upgrading Configurations

3.3.1. Upgrading Ambari Infra

If you have Ambari Solr installed, you must upgrade Ambari Infra after upgrading Ambari.

Steps

1. Make sure Ambari Infra services are stopped. From **Ambari Web**, browse to **Services > Ambari Infra** and select **Stop** from the **Service Actions** menu.
2. On every host in your cluster with an Infra Solr Client installed, run the following commands:

```
yum clean all
```

```
yum upgrade ambari-infra-solr-client
```

3. Execute the following command on all hosts running an Ambari Infra Solr Instance:

```
yum upgrade ambari-infra-solr
```

4. Start the Ambari Infra services.

   From **Ambari Web**, browse to **Services > Ambari Infra** select **Service Actions** then choose **Start**.

More Information

Ambari Infra

3.3.2. Upgrading Ambari Log Search

If you have Ambari Log Search installed, you must upgrade Ambari Log Search after upgrading Ambari.

Prerequisites

Before starting this upgrade, ensure the Ambari Infra components have been upgraded.

Steps

1. Make sure Ambari Log Search service is stopped. From **Ambari Web**, browse to **Services > Log Search** and select **Stop** from the **Service Actions** menu.
2. On every host in your cluster running a Log Feeder, run the following commands:

```
yum clean all
```

```
yum upgrade ambari-logsearch-logfeeder
```

3. Execute the following command on all hosts running the Log Search Server:

```
yum upgrade ambari-logsearch-portal
```

4. Start Log Search Service.

   From **Ambari Web**, browse to **Services > Log Search** select **Service Actions** then choose **Start**.

More Information

Upgrading Ambari Infra

3.3.3. Upgrading Ambari Metrics

Prerequisites

Upgrade to Ambari 2.6 and perform needed post-upgrade checks. Make sure all services are up and healthy.

Steps

1. Make sure Ambari Metrics service is stopped. From **Ambari Web**, browse to **Services > Ambari Metrics** and select **Stop** from the **Service Actions** menu.

2. On every host in your cluster running a Metrics Monitor, run the following commands:

```
yum clean all
```

```
yum upgrade ambari-metrics-monitor ambari-metrics-hadoop-sink
```

3. Execute the following command on all hosts running the Metrics Collector:

```
yum upgrade ambari-metrics-collector
```

4. Execute the following command on the host running the Grafana component:

```
yum upgrade ambari-metrics-grafana
```

5. Start Ambari Metrics Service.

   From **Ambari Web**, browse to **Services > Ambari Metrics** select **Service Actions** then choose **Start**.

   Updated Ambari Metrics Sink jars will be installed on all hosts and you must restart each service to pick up the latest sink implementations.

   Please wait to restart all services until after you have completed all applicable post-upgrade tasks, for example: HDFS, YARN, Kafka, HBase, Flume, Storm.

Next Steps

Restart services, only after you complete all applicable, post-upgrade tasks.

Note

New Ambari Metrics Sinks will not be activated until all services are restarted.

3.3.4. Upgrading Configurations

This section describes potential cluster configuration updates that may be required.

More Information

Upgrading Kerberos krb5.conf [11]

Upgrading Log Rotation Configuration [11]

3.3.4.1. Upgrading Kerberos krb5.conf

Ambari has added support for handling more than one KDC host . Only one kadmin host is supported by the Kerberos infrastructure. This required modifications for the **krb5.conf** template. In order for Ambari to properly construct the krb5.conf configuration file, make the following configuration change if your cluster meets all of these criteria:

- Kerberos is enabled and Ambari is configured for automated setup, and
- Ambari is managing the krb5.conf, and
- You **have modified** the krb5.conf template content from the default content. If you have not modified the default content, Ambari will automatically update the template content as part of upgrade and these configuration updates do not need to be applied manually.

If you meet all of the above criteria, you must update the **krb5.conf** template content found in **Services > Kerberos > Advanced**:

Original Template Entry	Updated Template Entry
admin_server = {{admin_server_host|default(kdc_host, True)}}	admin_server = {{admin_server_host|default(kdc_host_list[0]|trim(), True)}}
kdc = {{kdc_host}}	{% for kdc_host in kdc_host_list %} kdc = {{kdc_host|trim()}} {%- endfor -%}

More Information

Configure Ambari for Automated Setup

3.3.4.2. Upgrading Log Rotation Configuration

Ambari 2.6.0 provides s a simplified log rotation configuration. These changes will be made automatically during your next stack upgrade, but are not automatically made during the Ambari upgrade. After upgrading Ambari from version 2.x to 2.6.0, if you want to utilize

the simplified log rotation configuration, you must update configurations for all services in your cluster, using the following steps:

Steps

1. **ZooKeeper**

   a. In **Ambari Web**, browse to **ZooKeeper > Configs**.

   b. Scroll down to **Custom zookeeper-log4j**.

   c. In **Custom zookeeper-log4j**, click **Add Property**.

   d. In **Add Property**, type the following properties and values:

      zookeeper_log_max_backup_size=10

      zookeeper_log_number_of_backup_files=10

      For example:

   e. Click **Add**.

   f. Browse to **Advanced zookeeper-log4j**.

   g. In **Advanced zookeeper-log4j** *content section*, find and replace the following properties and values:

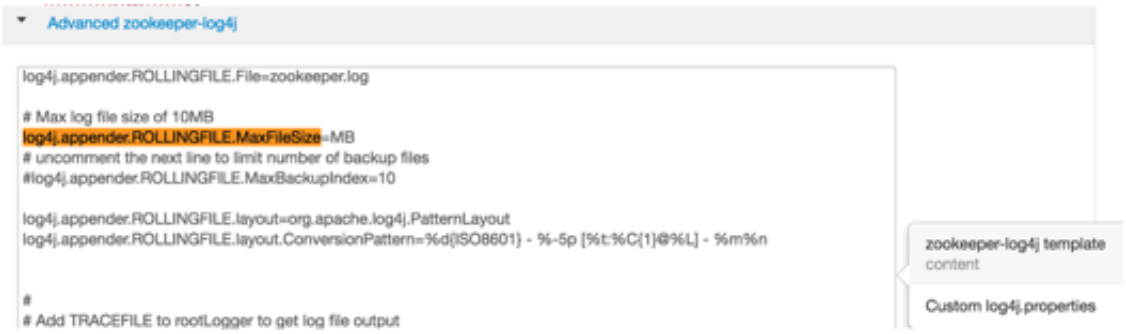
      Find: log4j.appender.ROLLINGFILE.MaxFileSize=<value>

      Replace:
      log4j.appender.ROLLINGFILE.MaxFileSize={{zookeeper_log_number_of_backup_files}}MB

      Find: #log4j.appender.ROLLINGFILE.MaxBackupIndex=<value>MB

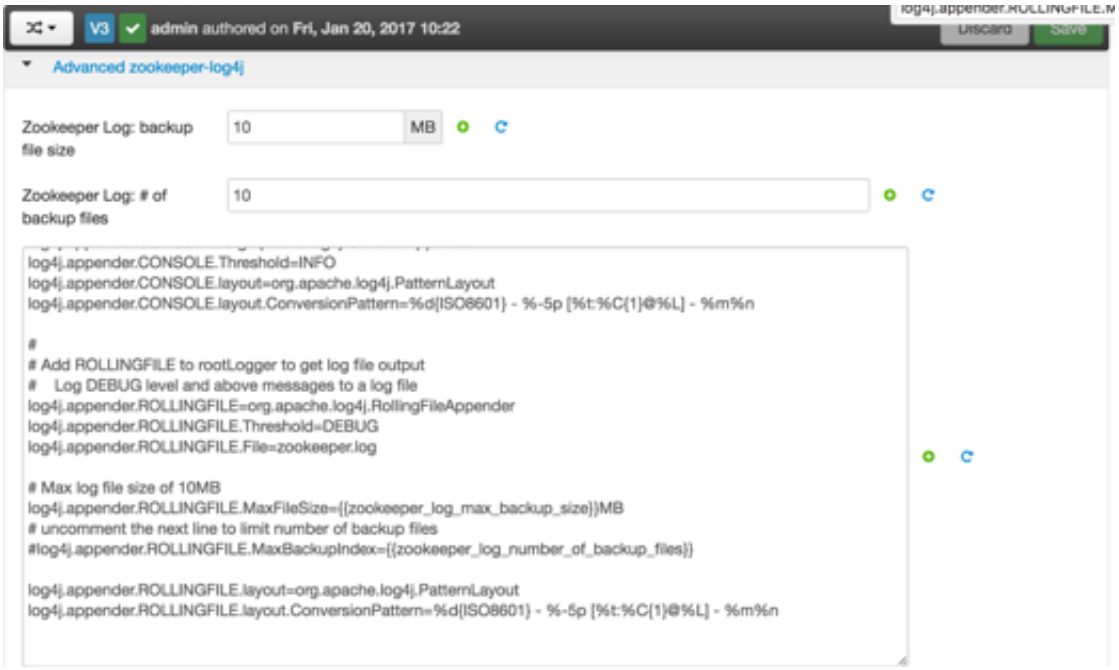
      Replace:
      #log4j.appender.ROLLINGFILE.MaxBackupIndex={{zookeeper_log_number_of_backup_files}}

      For example:



h. In **Configs**, click **Save**.

For example:



i. Restart **ZooKeeper**, as prompted.

2. **Kafka**

a. In **Ambari Web**, browse to **Kafka > Configs**.

b. Scroll down to **Custom Kafka-log4j**.

c. In **Custom Kafka-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

kafka_log_maxfilesize=256

kafka_log_maxbackupindex=20

controller_log_maxfilesize=256

controller_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced kafka-log4j**.

g. In **Advanced kafka-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.kafkaAppender=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.kafkaAppender.MaxFileSize = {{kafka_log_maxfilesize}}MB

Add: log4j.appender.kafkaAppender.MaxBackupIndex = {{kafka_log_maxbackupindex}}MB

Find: log4j.appender.controllerAppender=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.controllerAppender.MaxFileSize = {{controller_log_maxfilesize}}MB

Add: log4j.appender.controllerAppender.MaxBackupIndex = {{controller_log_maxbackupindex}}

h. In **Configs**, click **Save**.

i. Restart **Kafka**, as prompted.

3. **Knox**

a. In **Ambari Web**, browse to **Knox > Configs**.

b. Scroll down to **Custom gateway-log4j**.

c. In **Custom gateway-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

knox_gateway_log_maxfilesize=256

knox_gateway_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced gateway-log4j**.

g. In **Advanced gateway-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.drfa=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.drfa.MaxFileSize = {{knox_gateway_log_maxfilesize}}MB

Add: log4j.appender.drfa.MaxBackupIndex = {{knox_gateway_log_maxbackupindex}}

h. In **Configs**, click **Save**.

i. Restart **Knox**.

4. **Atlas**

a. In **Ambari Web**, browse to **Atlas > Configs > Advanced**.

b. Scroll down to **Custom Atlas-log4j**.

c. In **Custom Atlas-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

atlas_log_max_backup_size=256

atlas_log_number_of_backup_files=20

e. Click **Add**.

f. Browse to **Advanced atlas-log4j**.

g. In **Advanced atlas-log4j** *content section*, find and replace the following properties and values:

Find: <appender name="FILE" class="org.apache.log4j.DailyRollingFileAppender">

Add: <param name="MaxFileSize" value="{{atlas_log_max_backup_size}}MB" >

Add: <param name="MaxBackupIndex" value="{{atlas_log_number_of_backup_files}}" >

h. In **Configs**, click **Save**.

i. Restart **Atlas**, as prompted.

5. **Ranger**

a. In **Ambari Web**, browse to **Ranger > Configs > Advanced**.

b. Scroll down to **Custom admin-log4j**.

c. In **Custom admin-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

ranger_xa_log_maxfilesize=256

ranger_xa_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced admin-log4j**.

g. In **Advanced admin-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.xa_log_appender=org.apache.log4j.DailyRollingFileAppender

Add:
log4j.appender.xa_log_appender.MaxFileSize={{ranger_xa_log_maxfilesize}}MB

Add:
log4j.appender.xa_log_appender.MaxBackupIndex={{ranger_xa_log_maxbackupindex}}

h. Scroll down to **Custom usersync-log4j**.

i. In **Custom usersync-log4j**, click **Add Property**.

j. In **Add Property**, type the following properties and values:

ranger_usersync_log_maxfilesize=256

ranger_usersync_log_number_of_backup_files=20

k. Click **Add**.

l. Browse to **Advanced usersync-log4j**.

m. In **Advanced usersync-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.logFile=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.logFile.MaxFileSize = {{ranger_usersync_log_maxfilesize}}MB

Add: log4j.appender.logFile.MaxBackupIndex =
{{ranger_usersync_log_number_of_backup_files}}

n. Scroll down to **Custom tagsync-log4j**.

o. In **Custom tagsync-log4j**, click **Add Property**.

p. In **Add Property**, type the following properties and values:

ranger_tagsync_log_maxfilesize=256

ranger_tagsync_log_number_of_backup_files=20

q. Click **Add**.

r. Browse to **Advanced tagsync-log4j**.

s. In **Advanced tagsync-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.logFile=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.logFile.MaxFileSize = {{ranger_tagsync_log_maxfilesize}}MB

Add: log4j.appender.logFile.MaxBackupIndex = {{ranger_tagsync_log_number_of_backup_files}}

t. In **Configs**, click **Save**.

u. Restart **Ranger**, as prompted.

6. **Ranger-KMS**

a. In **Ambari Web**, browse to **Ranger-KMS > Configs > Advanced**.

b. Scroll down to **Custom kms-log4j**.

c. In **Custom kms-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

ranger_kms_log_maxfilesize=256

ranger_kms_log_maxbackupindex=20

ranger_kms_audit_log_maxfilesize=256

ranger_kms_audit_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced kms-log4j** .

g. In **Advanced kms-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.kms=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.kms.MaxFileSize = {{ranger_kms_log_maxfilesize}}MB

Add: log4j.appender.kms.MaxBackupIndex = {{ranger_kms_log_maxbackupindex}}

Find: log4j.appender.kms-audit=org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.kms-audit.MaxFileSize={{ranger_kms_audit_log_maxfilesize}}MB

Add: log4j.appender.kms-audit.MaxBackupIndex = {{ranger_kms_audit_log_maxbackupindex}}

h. In **Configs**, click **Save**.

i. Restart **Ranger-KMS**.

7. **HBase**

a. In **Ambari Web**, browse to **HBase > Configs > Advanced**.

b. Scroll down to **Custom hbase-log4j**.

c. In **Custom hbase-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

hbase_log_maxfilesize=256

hbase_log_maxbackupindex=20

hbase_security_log_maxfilesize=256

hbase_security_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced hbase-log4j** .

g. In **Advanced hbase-log4j** *content section*, find and replace the following properties and values:

Find: hbase.log.maxfilesize=<value>MB

Replace: hbase.log.maxfilesize={{hbase_log_maxfilesize}}MB

Find: hbase.log.maxbackupindex=<value>

Replace: hbase.log.maxbackupindex={{hbase_log_maxbackupindex}}

Find: hbase.security.log.maxfilesize=<value>MB

Replace: hbase.security.log.maxfilesize={{hbase_security_log_maxfilesize}}MB

Find: hbase.security.log.maxbackupindex=<value>

Replace:
hbase.security.log.maxbackupindex={{hbase_security_log_maxbackupindex}}

h. In **Configs**, click **Save**.

i. Restart **HBase**.

8. **Hive**

a. In **Ambari Web**, browse to **Hive > Configs > Advanced**.

b. Scroll down to **Custom hive-log4j**.

c. In **Custom hive-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

   hive_log_maxfilesize=256

   hive_log_maxbackupindex=30

e. Click **Add**.

f. Browse to **Advanced hive-log4j** .

g. In **Advanced hive-log4j** *content section*, find and replace the following properties and values:

   Find: #log4j.appender.DRFA.MaxBackupIndex=<value>

   Replace: #log4j.appender.DRFA.MaxBackupIndex= {{hive_log_maxbackupindex}}

   Find: log4j.appender.DRFA.MaxBackupIndex=<value>

   Add: log4j.appender.DRFA.MaxFileSize = {{hive_log_maxfilesize}}MB

h. Scroll down to **Custom llap-daemon-log4j**.

i. In **Custom llap-daemon-log4j**, click **Add Property**.

j. In **Add Property**, type the following properties and values:

   hive_llap_log_maxfilesize=256

   hive_log_maxbackupindex=240

k. Click **Add**.

l. Browse to **llap-daemon-log4j**.

m. In **llap-daemon-log4j** *content section*, find and replace the following properties and values:

   Find: property.llap.daemon.log.maxfilesize =<value>MB

   Replace: property.llap.daemon.log.maxfilesize = {{hive_llap_log_maxfilesize}}MB

   Find: property.llap.daemon.log.maxbackupindex=<value>

   Replace:
   property.llap.daemon.log.maxbackupindex={{hive_llap_log_maxbackupindex}}

n. Scroll down to **Custom webhcat-log4j**.

o. In **Custom webhcat-log4j**, click **Add Property**.

p. In **Add Property**, type the following properties and values:

webhcat_log_maxfilesize=256

webhcat_log_maxbackupindex=240

q. Click **Add**.

r. Browse to **webhcat-log4j** .

s. In **webhcat-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.standard = org.apache.log4j.DailyRollingFileAppender

Add: log4j.appender.standard.MaxFileSize = {{webhcat_log_maxfilesize}}MB

Add: log4j.appender.standard.MaxBackupIndex = {{webhcat_log_maxbackupindex}}

t. In **Configs**, click **Save**.

u. Restart **Hive**, as prompted.

9. **Storm**

a. In **Ambari Web**, browse to **Storm > Configs**.

b. Scroll down to **Custom cluster-log4j property**.

c. In **Custom cluster-log4j property**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

storm_a1_maxfilesize=100

storm_a1_maxbackupindex=9

e. Click **Add**.

f. Browse to **Advanced storm-cluster-log4j** .

g. In **Advanced storm-cluster-log4j** *content section*, find and replace the following properties and values:

Find: In RollingFile="A1"<SizeBasedTriggeringPolicy size="<value>MB"/>

Replace: <SizeBasedTriggeringPolicy size="{{storm_a1_maxfilesize}}MB"/>

Find: In RollingFile="A1"<DefaultRolloverStrategy max="<value>"/>

Replace: <DefaultRolloverStrategy max="{{storm_a1_maxbackupindex}}"/>

h. Scroll down to **Custom worker-log4j property**.

i. In **Custom worker-log4j property**, click **Add Property**.

j. In **Add Property**, type the following properties and values:

storm_wrkr_a1_maxfilesize=100

storm_wrkr_a1_maxbackupindex=9

storm_wrkr_out_maxfilesize=100

storm_wrkr_out_maxbackupindex=4

storm_wrkr_err_maxfilesize=100

storm_wrkr_err_maxbackupindex=4

k. Click **Add**.

l. Browse to **Advanced storm-worker-log4j** .

m. In **Advanced storm-worker-log4j** *content section*, find and replace the following properties and values:

Find: In RollingFile="A1"<SizeBasedTriggeringPolicy size="<value> MB"/>

Replace: <SizeBasedTriggeringPolicy size="{{storm_wrkr_a1_maxfilesize}} MB"/>

Find: In RollingFile="A1"<DefaultRolloverStrategy max="<value>"/>

Replace: <DefaultRolloverStrategy max="{{storm_wrkr_a1_maxbackupindex}}"/>

Find: In RollingFile="STDOUT"<SizeBasedTriggeringPolicy size="<value>" MB/>

Replace: <SizeBasedTriggeringPolicy size="{{storm_wrkr_out_maxfilesize}} MB"/>

Find: In RollingFile="STDOUT"<DefaultRolloverStrategy max="<value>"/>

Replace: <DefaultRolloverStrategy max="{{storm_wrkr_out_maxbackupindex}}"/>

Find: In RollingFile="STDERR"<SizeBasedTriggeringPolicy size="<value>" MB/>

Replace: <SizeBasedTriggeringPolicy size="{{storm_wrkr_err_maxfilesize}} MB"/>

Find: In RollingFile="STDOUT"<DefaultRolloverStrategy max="<value>"/>

Replace: <DefaultRolloverStrategy max="{{storm_wrkr_err_maxbackupindex}}"/>

n. In **Configs**, click **Save**.

o. Restart **Storm**, as prompted.

10. **Falcon**

a. In **Ambari Web**, browse to **Falcon > Configs**.

b. Scroll down to **Custom falcon-log4j**.

c. In **Custom falcon-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

falcon_log_maxfilesize=256

falcon_log_maxbackupindex=20

falcon_security_log_maxfilesize=256

falcon_security_log_maxbackupindex=20

e. Click **Add**.

f. Browse to **Advanced falcon-log4j** .

g. In **Advanced falcon-log4j** *content section*, find and replace the following properties and values:

Find: <appender name="FILE" class="org.apache.log4j.DailyRollingFileAppender">

Add: <param name="MaxFileSize" value="{{falcon_log_maxfilesize}}MB" />

Add: <param name="MaxBackupIndex" value="{{falcon_log_maxbackupindex}}" />

Find: <appendername="SECURITY"class="org.apache.log4j.DailyRollingFileAppender">

Add: <param name="MaxFileSize" value="{{falcon_security_log_maxfilesize}}MB"/>

Add: <param name="MaxBackupIndex"
value="{{falcon_security_log_maxbackupindex}}"/>

h. In **Configs**, click **Save**.

i. Restart **Falcon**.

11. **Oozie**

a. In **Ambari Web**, browse to **Oozie > Configs**.

b. Scroll down to **Custom oozie-log4j**.

c. In **Custom oozie-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

oozie_log_maxhistory=720

e. Click **Add**.

f. Browse to **Advanced oozie-log4j** .

g. In **Advanced oozie-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.oozie.RollingPolicy.MaxHistory=720

Replace: log4j.appender.oozie.RollingPolicy.MaxHistory={{oozie_log_maxhistory}}

h. In **Configs**, click **Save**.

i. Restart **Oozie**, as prompted.

12. **HDFS**

a. In **Ambari Web**, browse to **HDFS > Configs > Advanced**.

b. Scroll down to **Custom HDFS-log4j**.

c. In **Custom HDFS-log4j**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

hadoop_security_log_max_backup_size = 256

hadoop_security_log_number_of_backup_files = 20

hadoop_log_max_backup_size = 256

hadoop_log_number_of_backup_files = 10

e. Click **Add**.

f. Browse to **Advanced hdfs-log4j** .

g. In **Advanced hdfs-log4j** *content section*, find and replace the following properties and values:

Find: hadoop.security.log.maxfilesize= <value>MB

Replace:
hadoop.security.log.maxfilesize={{hadoop_security_log_max_backup_size}}MB

Find: hadoop.security.log.maxbackupindex=20

Replace:
hadoop.security.log.maxbackupindex={{hadoop_security_log_number_of_backup_files}}

Find: log4j.appender.RFA.MaxFileSize=<value>MB

Replace: log4j.appender.RFA.MaxFileSize={{hadoop_log_max_backup_size}}MB

Find: log4j.appender.RFA.MaxBackupIndex=<value>

Replace:
log4j.appender.RFA.MaxBackupIndex={{hadoop_log_number_of_backup_files}}

h. In **Configs**, click **Save**.

i. Restart **HDFS**, as prompted.

13.**YARN**

a. In **Ambari Web**, browse to **YARN > Configs > Advanced**.

b. Scroll down to **Custom YARN-log4j property**.

c. In **Custom YARN-log4j property**, click **Add Property**.

d. In **Add Property**, type the following properties and values:

yarn_rm_summary_log_max_backup_size=256

yarn_rm_summary_log_number_of_backup_files=20

e. Click **Add**.

f. Browse to **Advanced yarn-log4j** .

g. In **Advanced yarn-log4j** *content section*, find and replace the following properties and values:

Find: log4j.appender.RMSUMMARY.MaxFileSize=<value>MB

Replace:
log4j.appender.RMSUMMARY.MaxFileSize={{yarn_rm_summary_log_max_backup_size}}MB

Find: log4j.appender.RMSUMMARY.MaxBackupIndex=<value>

Replace:
log4j.appender.RMSUMMARY.MaxBackupIndex={{yarn_rm_summary_log_number_of_backup_files}}

h. In **Configs**, click **Save**.

i. Restart **YARN**, as prompted.

3.3.5. Upgrading SmartSense

If your cluster includes the SmartSense service, you must upgrade it after upgrading Ambari.

More Information

Upgrading SmartSense

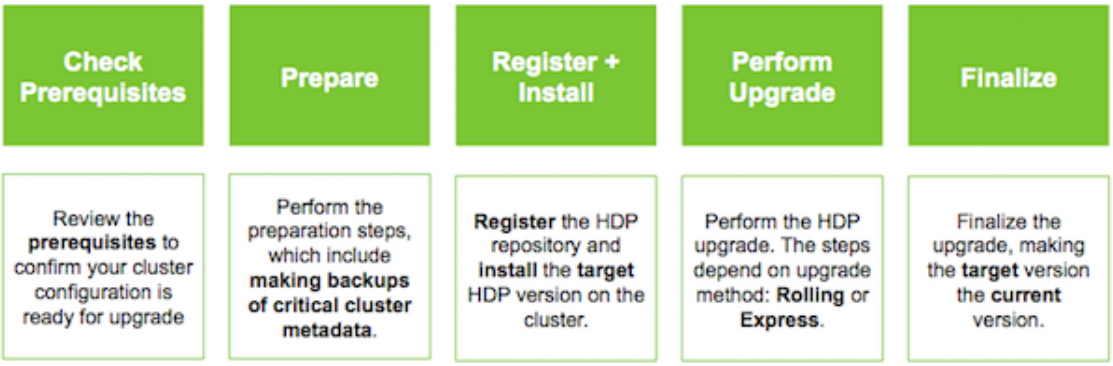
Next Steps

Restart services.

4. Upgrading HDP

There are different HDP upgrade options based on your current HDP version and the target HDP version. This section describes the different upgrade options, their prerequisites, and the overall process. You are **strongly encouraged** to read completely through this entire document before starting the upgrade process, to that you understand the interdependencies and order of the steps. It is **highly recommended** you validate these steps in a test environment to adjust and account for any special configurations for your cluster.

The high-level process for performing an HDP upgrade is as follows:



Ambari will guide you through the steps required to upgrade HDP. Make sure Ambari and the cluster are healthy, operating normally and all service checks are passing.

Note

Be sure to review the available HDP upgrade scenarios below. It is **strongly recommended** that you **first upgrade to Ambari 2.6.0** before upgrading HDP unless otherwise noted. After upgrading Ambari, be sure the cluster is operating normally and service checks are passing prior to attempting an HDP upgrade.

There are two methods for upgrading HDP with Ambari: **Rolling Upgrade** and **Express Upgrade**.

- A **Rolling Upgrade** orchestrates the HDP upgrade in an order that is meant to preserve cluster operation and minimize service impact during upgrade. This process has more stringent prerequisites (particularly regarding cluster high availability configuration) and can take longer to complete than an Express Upgrade.
- An **Express Upgrade** orchestrates the HDP upgrade in an order that will incur cluster downtime but with less stringent prerequisites.

The following table describes which upgrade option is available based on the current and target HDP version combinations. It is important to note that the HDP Stack version is based on the following format: **major.minor.maintenance.patch-build#**. A "minor" release is a release that increments the second-digit. A "maintenance" release is one that increments the third-digit. This terminology is used in the following table to describe the different upgrade paths. For example: an upgrade from HDP 2.5 to HDP 2.6, is a "Minor Upgrade". An upgrade from HDP 2.6.1 to HDP 2.6.2, is a "Maintenance" Upgrade.

Table 4.1. HDP Upgrade Options for IBM-PPC

Upgrade Path	Current HDP*	Target HDP*	Rolling Upgrade	Express Upgrade
Maintenance	HDP 2.6.1.0	HDP 2.6.2.0	✅	✅

More Information

Preparing to Upgrade Ambari and HDP

Rolling Upgrade Prerequisites [27]

4.1. Prerequisites

To perform an HDP upgrade using Ambari, your cluster must meet the following prerequisites, whether you intend to perform a Rolling or Express upgrade. Meeting these prerequisites is essential for Ambari to know the cluster is in a healthy operating mode and can successfully manage the upgrade process. Your cluster must meet additional prerequisites for a Rolling Upgrade, which ensure that the cluster is configured properly to support the rolling upgrade orchestration process.

HDP Version — All hosts must have the target HDP version installed.

Disk Space — Be sure to have adequate space on `/usr/hdp` for the target HDP version. Each complete install of an HDP version will occupy about 2.5 GB of disk space.

Ambari Agent Heartbeats — All Ambari Agents must be communicating and heartbeating to Ambari Server. Any hosts that are not heartbeating must be in Maintenance Mode.

Host Maintenance Mode — The following two scenarios are checked:

- Any hosts in Maintenance Mode must not be hosting any Service Master Components.
- Any host in Maintenance Mode that is not hosting Master Components is allowed but you will receive a warning. You can proceed with your upgrade but these hosts will not be upgraded and **before** you can finalize the upgrade, you must delete the hosts from the cluster.

Service Maintenance Mode — No Services can be in Maintenance Mode.

Services Started — All Services must be started.

Service Checks	All Service Checks must pass. Be sure to run **Service Actions > Run Service Check** on all services (and remediate if necessary) prior to attempting an HDP upgrade.
Kerberos - Kafka	**Ranger Policy Export/Import DB Requirements** HDP-2.6 and higher versions include Ranger policy import/export. The mininum database requirements for this feature are as follows: • MariaDB: 10.1.16+ • MySQL: 5.6.x+ • Oracle: 11gR2+ • PostgreSQL: 8.4+ • MS SQL: 2008 R2+

More Information

Rolling Upgrade Prerequisites [27]

Register and Install Target Version [32]

Remove Service

4.1.1. Rolling Upgrade Prerequisites

To perform a **Rolling Upgrade**, your cluster must meet the following prerequisites. If your cluster does not meet these upgrade prerequisites, you can consider an **Express Upgrade**.

HDFS

NameNode HA	NameNode HA must be enabled and working properly. From Ambari Web, the NameNodes, ZKFailoverControllers and JournalNodes must be “green” and showing no alerts.
NameNode Truncate	In HDP 2.2, truncate **must** be disabled and this prerequisite is **REQUIRED**. The hdfs-site configuration dfs.allow.truncate, property is set to true if truncate is enabled. If the property is not set or is false, truncate is disabled. In HDP 2.3 and later, truncate is always enabled, which is acceptable. Therefore, this check is not applicable and will not be performed with HDP 2.3 maintenance upgrades.

YARN

Timeline Service State Recovery	YARN state recovery should be enabled. Check the **Services > YARN > Configs > Advanced** property yarn.timeline-service.recovery.enabled is set to true.

ResourceManager Work Preserving Recovery	YARN work preserving recovery should be enabled. Check the **Services > YARN > Configs > Advanced** property yarn.resourcemanager.work-preserving-recovery.enabled is set to true.
ResourceManager HA	ResourceManager HA should be enabled to prevent a disruption in service during the upgrade. This prerequisite is **OPTIONAL**.
MapReduce2	
MapReduce Distributed Cache	MapReduce should reference Hadoop libraries from the distributed cache in HDFS.
JobHistory State Recovery	JobHistory state recovery should be enabled. Check the following: **Services > MapReduce > Configs > Advanced** property mapreduce.jobhistory.recovery.enable is set to true . Once enabled, dependent properties for mapreduce.jobhistory.recovery.store.class and mapreduce.jobhistory.recovery.store.leveldb.path should also be set.
Encrypted Shuffle	If encrypted shuffle has been enabled, the `ssl-client.xml` file must be copied to `/etc/hadoop/conf/secure` directory on each node in the cluster.
Tez	
Tez Distributed Cache	Tez should reference Hadoop libraries from the distributed cache in HDFS. Check the **Services > Tez > Configs** configuration and confirm tez.lib.uris is set with the path in HDFS for the `tez.tar.gz` file.
Hive	
Hive Dynamic Service Discovery	HiveServer2 dynamic service discovery should be configured for Rolling Upgrade.
Hive Client Retry	Hive client retry properties must be configured. Review the **Services > Hive > Configs** configuration and confirm hive.metastore.failure.retries and hive.metastore.client.connect.retry.delay are specified.
Multiple Hive Metastore	Multiple Hive Metastore instances are recommended for Rolling Upgrade. This ensures that there is at least one Hive Metastore running during the upgrade process. Additional Hive Metastore instances can be added through Ambari. This prerequisite is **OPTIONAL**.
Oozie	

Oozie Client Retry	Oozie client retry properties must be configured. Review the **Services > Oozie > Configs > oozie-env** configuration and confirm "export OOZIE_CLIENT_OPTS="${OOZIE_CLIENT_OPTS} -Doozie.connection.retry.count=<number of retries>" is specified.

More Information

Configuring NameNode High Availability.

ResourceManager High Availability.

YARN Resource Management

4.2. Prepare to Upgrade

Make sure that you have review and completed all prerequisites described in previous chapters.

It is **strongly** recommended that you perform backups of your databases before beginning upgrade.

- Ambari database
- Hive Metastore database
- Oozie Server database
- Ranger Admin database
- Ranger Audit database

Important

If you use MySQL 5.6, you must use the Inno DB engine.

If your current MySQL engine is MyISAM, you must migrate to Inno DB before upgrading to Ambari 2.6.0

- Export your current, MySQL (MyISAM) data.
- Drop the MySQL (MyISAM) database schema.
- Create a MySQL (Inno DB) database.
- Import your data into the MySQL (Inno DB) database instance.

For example:

```
mysqldump -U ambari -p ambari > /tmp/ambari.original.mysql
```

```
cp /tmp/ambari.original.mysql /tmp/ambari.innodb.mysql
```

```
sed -ie 's/MyISAM/INNODB/g' /tmp/ambari.innodb.mysql
```

```
mysql -u ambari -p ambari

DROP DATABASE ambari;

CREATE DATABASE ambari;

mysql -u ambari "-pbigdata" --force ambari < /tmp/ambari.innodb.
mysql
```

Please contact Hortonworks customer support if you have issues running your cluster using MySQL 5.6.

Checkpoint HDFS

1. Perform the following steps on the NameNode host. If you are configured for NameNode HA, perform the following steps on the Active NameNode. You can locate the Active NameNode from **Ambari Web > Services > HDFS** in the **Summary** area.

2. Check the NameNode directory to ensure that there is no snapshot of any prior HDFS upgrade. Specifically, using Ambari Web, browse to **Services > HDFS > Configs**, and examine the dfs.namenode.name.dir in the NameNode Directories property. Make sure that only a `/current` directory and no `/previous` directory exists on the NameNode host.

3. Create the following log and other files. Creating these logs allows you to check the integrity of the file system after the Stack upgrade.

   As the HDFS user,

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

   run the following (where <HDFS_USER> is the HDFS Service user, for example, hdfs):

   - 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
     ```

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

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

   - **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
     ```

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

4. Save the namespace. As the HDFS user, " su -l <HDFS_USER> ", you must put the cluster in Safe Mode.

   ```
   hdfs dfsadmin -safemode enter
   ```

```
hdfs dfsadmin -saveNamespace
```

Note

In a highly-available NameNode configuration, the command

```
hdfs dfsadmin -saveNamespace
```

sets a checkpoint in the first NameNode specified in the configuration, in `dfs.ha.namenodes.[nameserviceID]`.

You can also use the

```
dfsadmin -fs
```

option to specify which NameNode to connect. For example, to force a checkpoint in NameNode2:

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

5. Copy the checkpoint files located in `${dfs.namenode.name.dir}/current` into a backup directory.

Note

In a highly-available NameNode configuration, the location of the checkpoint depends on where the

```
saveNamespace
```

command is sent, as defined in the preceding step.

6. Store the layoutVersion for the NameNode located at

   `${dfs.namenode.name.dir}/current/VERSION`, into a backup directory where

   `${dfs.namenode.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.

7. As the HDFS user, " su -l <HDFS_USER> ", take the NameNode out of Safe Mode.

```
hdfs dfsadmin -safemode leave
```

8. Finalize any prior HDFS upgrade, if you have not done so already. As the HDFS user, " su -l <HDFS_USER> ", run the following:

```
hdfs dfsadmin -finalizeUpgrade
```

Important

In HDP-2.5, Audit to DB is no longer supported in Apache Ranger. If you are upgrading from HDP-2.3 or HDP-2.4 to HDP-2.5, you should use Apache Solr for Ranger audits. You should also migrate your audit logs from DB to Solr.

Next Steps

Register and Install Target Version.

More Information

Getting Ready to Upgrade Ambari and HDP

Using Apache Solr for Ranger audits

Migrate your audit logs from DB to Solr

4.3. Register and Install Target Version

This section describes the steps to register the software repositories with Ambari for the new target version (i.e. the version you will be upgrading to) and how to install the software on all hosts before performing the upgrade.

Register Target Version

Steps

1. Log in to Ambari.
2. Browse to **Admin > Stack and Versions**.
3. Click the **Versions** tab. You see the version currently running, marked as **Current**.

Note

The full version depends on the HDP version you are actually running. For example, if you are currently running the HDP 2.6.1.0 release, you would see something like **HDP-2.6.10-1479** as the full version number.

4. Click **Manage Versions**.
5. Proceed to register a new version by clicking **+ Register Version**.
6. Select the software version and method of delivery for your cluster.
   a. **Choose HDP Stack**.

      The available HDP versions are shown in tabs. When you select a tab, Ambari attempts to discover what specific version of that HDP Stack is available. That list is shown in a drop-down. For that specific version, the available Services are displayed, with their Versions shown in the table.

   b. **Choose HDP Version**.

      If Ambari has access to the Internet, the specific Versions will be listed as options in the DROPDOWN. If you have a Version Definition File for a version that is not listed, you can click **Add Version...** and upload the VDF file. In addition, a **Default Version Definition** is also included in the list if you do not have Internet access or are not sure

which specific version to install. If you choose the **Default Version Definition**, you must enter a "two-digit Version Number" in the **Name** input field.

c. **Choose Repository Delivery Method**.

Using a Public Repository requires Internet connectivity. Using a Local Repository requires you have configured the software in a repository available in your network. To use the public software repositories, see the list of available HDP Repositories for each OS. Or, if you are using a local repository, enter the Base URLs for the local repository you have created.

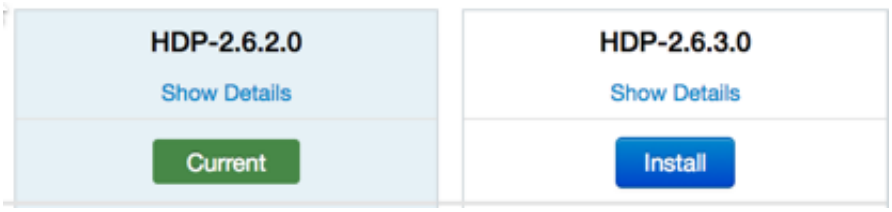
7. Click **Save**.
8. Click **Go To Dashboard**, and browse back to **Stack and Versions > Versions**.
9. You will see the version currently running, marked **Current**, and the version you just registered **HDP-2.6.3.0** displaying an **Install** button.

Next Steps

Install Target Version

Steps

1. Log in to Ambari.
2. Browse to **Admin > Stack and Versions**.
3. Click the **Versions** tab.



4. Click **Install** and click **OK** to confirm.

The Install version operation starts. This installs the target version on all hosts in the cluster. You can monitor the progress of the install by clicking the **Installing** link.

Note

If you have Apache Ranger on your cluster, confirm that the `ambari-infra-solr-client` package is installed on the Ranger Admin host. The **Ambari Infra** service contains the default Solr instance used by Apache Ranger.

When the install completes, the **Upgrade** button replaces the **Install** button.

Next Steps

Perform the Upgrade.

More Information

Managing Versions

Using a Local Repository

Ambari Infra

4.4. Perform the Upgrade

To upgrade your HDP version:

Steps

1. Log in to Ambari.
2. Browse to **Admin > Stack and Versions**.
3. Click the **Versions** tab.

   The registered and installed target HDP version displays.
4. Click **Perform Upgrade** on the target version.

   Based on your current HDP version and the target HDP version, Ambari performs a set of prerequisite checks to determine if you can perform a rolling or an express upgrade. A dialog displays the options available.
5. Select your upgrade method: **Rolling Upgrade** or **Express Upgrade**.

   Advanced options are also available.

Skip all Service Check failures	Ambari automatically skips any Service Check failures and completes the task without requiring user actions. After all the Service Checks have run in a task, you see a summary of the failures and options to continue the upgrade or pause.
Skip all Slave Component failures	Ambari automatically skips any Slave Component failures and completes the task of upgrading Slave components without requiring user actions. After all Slave Components have been upgraded, you see a summary of the failures and options to continue the upgrade or pause.

6. Click **Proceed**.

   Based on your option choice, proceed with the upgrade:

Next Steps

Perform Rolling Upgrade [35]

Perform Express Upgrade [37]

More Information

Register and Install Target Version [32]

Prerequisites [26]

4.4.1. Perform Rolling Upgrade

1. Ambari checks that your cluster meets prerequisites. A dialog displays the results:

   a. If any *required* prerequisites are not met, the result displays an error.

      You cannot proceed with the upgrade until you make the appropriate corrections and return to Perform Upgrade again.

   b. If any *optional* prerequisites are not met, the result displays a warning.

      You may proceed with the upgrade.

   c. Ambari displays a list of configuration changes that occur during the upgrade.

2. When the prerequisite checks complete, the upgrade starts. The time required to perform the upgrade depends on many factors. As part of the upgrade process, each component in the cluster restarts in a serial fashion. The stop/start times contribute to the total upgrade time.

3. The upgrade process includes the following stages. Some stages require that you complete an action during normal operation.

   If any stage fails, the upgrade stops and prompts you for action.

Stage	Description	Action Required
Prepare Backups	This step prompts you to confirm that you have taken proper backups before proceeding.	You must acknowledge the prompt for database backups.
ZooKeeper	All ZooKeeper servers are upgraded and restarted.	None
Ranger	Ranger Admin and UserSync servers are upgraded and restarted.	None. If Ranger Admin does not function after the upgrade completes, run the upgrade scripts manually, then Retry Upgrading Ranger.
Core Masters	This stage upgrades the master components of core services. This includes JournalNodes & NameNodes (HDFS), HistoryServer (MapReduce2), ResourceManager & ATS (YARN) and HBase Masters (HBase).	None
All Service Checks	All Service Checks are performed against the cluster.	Any service check that fail prompt you to **Ignore and Continue**, **Downgrade** or **Retry**. If you selected the **Skip all Service Check failures** option, you will only be prompted when all Service Checks complete.
Core Slaves	This stage upgrades the slave components of core services. This includes DataNodes (HDFS), RegionServers (HBase) and NodeManagers (YARN). This is done in two batches: 20% of the slaves first, then the remaining slaves.	After the first 20% batch completes, you are prompted to verify the cluster is operating correctly.

Stage	Description	Action Required
All Service Checks	All Service Checks are performed against the cluster.	Any service check that fail prompt you to **Ignore and Continue**, **Downgrade** or **Retry**. If you selected the **Skip all Service Check failures** option, you will only be prompted when all Service Checks complete.
Spark	The Spark Job History Server and clients are upgraded.	None
Oozie	The Oozie Server and clients are upgraded.	None
Falcon	The Falcon Server and clients are upgraded.	None
Client Components	All remaining clients are upgraded.	None
All Service Checks	All Service Checks are performed against the cluster.	Any service check that fails prompts you to **Ignore and Continue**, **Downgrade** or **Retry**. If you selected the **Skip all Service Check failures** option, you will only be prompted when all Service Checks complete.
Kafka	The Kafka Brokers are upgraded.	None
Knox	The Knox Gateways are upgraded.	None
Storm	Storm does not support rolling upgrade from a previous HDP version to HDP-2.5.	The rolling upgrade process prompts you to stop Storm topologies, perform the upgrade, and redeploy your topologies.
Slider	The Slider clients are upgraded.	None
Flume	The Flume agents are upgraded.	None
Finalize Upgrade Pre-Check	Checks if any hosts were not upgraded, either because the host was in Maintenance Mode, or one or more components on the host failed to upgrade (and were skipped).	Click the list that displays **# hosts** for details on the hosts (and their components) that are not upgraded. You can **Pause Upgrade**, delete the hosts and return to finalize.
Finalize	The component upgrades are complete. You are prompted to **Finalize**. Finalizing completes the upgrade process and saves the cluster state.	Prompted to Finalize, Finalize Later or Downgrade.

4. When the rolling upgrade stages complete, may choose to **Finalize** the upgrade, to **Finalize Later** or to **Downgrade**. Finalizing later gives you a chance to perform more validation on the cluster. Downgrade moves the cluster version back to the previous version (basically: reverses the upgrade process stages). **Once finalized, you cannot downgrade back to the previous version**.

Note

If you choose to finalize later, both versions will be listed on the Stack and Versions tab with the starting version displaying as Current. It is not until you finalize that Ambari makes the target version the current version. Also, until you finalize, you will not be able to perform operational changes to the cluster (such as move components, change configurations, etc).

5. Click **Finalize** to complete the rolling upgrade process.

More Information

Rolling Upgrade Prerequisites [27]

Retry Ranger Upgrade

4.4.2. Perform Express Upgrade

1. Ambari checks that your cluster meets prerequisites. A dialog displays the results:

   a. If any *required* prerequisites are not met, the result displays an error.

      You cannot proceed with the upgrade until you make the appropriate corrections and return to Perform Upgrade again.

   b. If any *optional* prerequisites are not met, the result displays a warning.

      You may proceed with the upgrade.

   c. Ambari displays a list of configuration changes that occur during the upgrade.

2. When the prerequisite checks complete, the upgrade starts. The time required to perform the upgrade depends on many factors. As part of the upgrade process, each component in the cluster restarts in a serial fashion. The stop/start times contribute to the total upgrade time.

3. The upgrade process includes the following stages. Some stages require that you complete an action during normal operation.

   If any stage fails, the upgrade stops and prompts you for action.

Stage	Description	Action Required
Prepare Upgrade	You should stop all YARN queues, all long-running applications on Slider, and deactivate & kill all running Storm topologies.	Perform the actions to prepare for the upgrade.
Stop Components for High-Level Services	This will stop all components for High-Level Services. This includes all master components **except** those of HDFS, HBase, ZooKeeper and Ranger.	None
Perform Backups	This step prompts you to confirm that you have taken proper backups before proceeding.	You must acknowledge the prompt for database backups.
Stop Components for Core Service	Stops all components with HDFS, HBase, ZooKeeper and Ranger.	None
Update Target Stack	Updates the stack version in Ambari to the target version. There is no downgrade past this point.	None
Update Service Configs	Updates (i.e. transfers or replaces) any configurations that are necessary for the upgrade.	None
Restart Components	Restarts all core components such as ZooKeeper, Ranger, HDFS, YARN, MapReduce2 and various Clients (Tez, Pig, Sqoop).	None
All Service Checks	All Service Checks are performed against the cluster.	Any service check that fails prompts you to **Ignore and Continue**, **Downgrade** or **Retry**. If you selected the **Skip all Service Check failures** option, you will only be prompted when all Service Checks complete.
Restart Components	Restarts the remaining components such as Oozie, Falcon, Hive, Spark and others.	None

Stage	Description	Action Required
Set Version on All Hosts	Sets the HDP version on all hosts to the target HDP version.	None
Finalize Upgrade Pre-Check	Checks if any hosts were not upgraded, either because the host was in Maintenance Mode, or one or more components on the host failed to upgrade (and were skipped).	Click the list that displays **# hosts** for details on the hosts (and their components) that are not upgraded. You can **Pause Upgrade**, delete the hosts and return to finalize.
Finalize Upgrade	The component upgrades are complete. You are presented the option to Finalize, which when selected, completes the upgrade process + saves the cluster state.	Prompted to Finalize or Finalize Later or Downgrade.

4. When the rolling upgrade stages complete, may choose to **Finalize** the upgrade, to **Finalize Later** or to **Downgrade**. Finalizing later gives you a chance to perform more validation on the cluster. Downgrade moves the cluster version back to the previous version (basically: reverses the upgrade process stages). **Once finalized, you cannot downgrade back to the previous version**.

Note

If you choose to finalize later, both versions will be listed on the Stack and Versions tab with the starting version displaying as Current. It is not until you finalize that Ambari makes the target version the current version. Also, until you finalize, you will not be able to perform operational changes to the cluster (such as move components, change configurations, etc).

5. Click **Finalize** to complete the express upgrade process.

More Information

Register and Install Target Version [32]

Prerequisites [26]

4.5. Post-upgrade Tasks

Post-upgrade Tasks for Ranger Usersnyc

For HDP-2.6 and higher, Ranger Usersync supports incremental sync for LDAP sync. For users upgrading with LDAP sync, the `ranger.usersync.ldap.deltasync` property (found in Ranger User Info with the label "Incremental Sync") will be set to `false`. This property is set to `true` in new installs of Ranger in HDP-2.6 and higher.

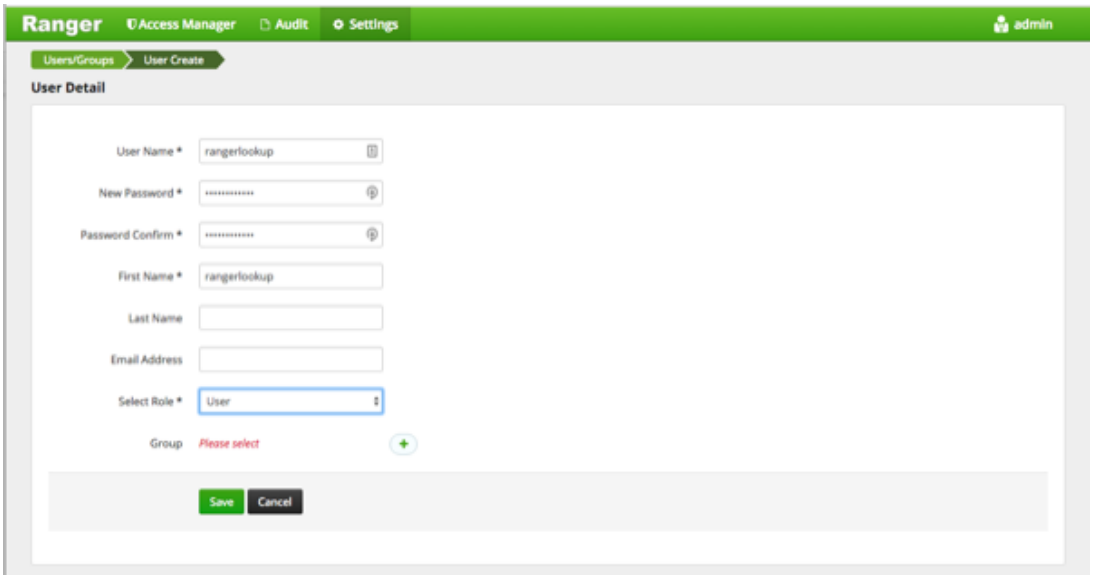
Post-upgrade Tasks for Ranger with Kerberos

1. If you have not already done so, you should migrate your audit logs from DB to Solr.

2. After successfully upgrading, you must regenerate keytabs. Select the **Only regenerate keytabs for missing hosts and components** check box, confirm the keytab regeneration, then restart all required services.

3. Log in to the Ranger Admin UI and select **Settings > Users/Groups**. Select the **Users** tab, then click **Add New User**. On the **User Detail** page, enter the short-name of the

principal ranger.lookup.kerberos.principal (if that user does not already exist) in the **User Name** box. For example, if the lookup principal is rangerlookup/_HOST@${realm}, enter rangelookup as the user name. Enter the other settings shown below, then click **Save** to create the user. Add the new rangerlookup user in Ranger policies and services. You can use **Test Connection** on the **Services** pages to chcck the connection.



Additional Steps for Ranger HA with Kerberos

If Ranger HA (High Availability) is set up along with Kerberos, perform the following additional steps:

1. Use SSH to connect to the KDC server host. Use the

   ```
   kadmin.local
   ```

   command to access the Kerberos CLI, then check the list of principals for each domain where Ranger Admin and the load-balancer are installed.

   ```
   kadmin.local
   kadmin.local: list_principals
   ```

   For example, if Ranger Admin is installed on <host1> and <host2>, and the load-balancer is installed on <host3>, the list returned should include the following entries:

   `HTTP/ <host3>@EXAMPLE.COM HTTP/ <host2>@EXAMPLE.COM HTTP/ <host1>@EXAMPLE.COM`

   If the HTTP principal for any of these hosts is not listed, use the following command to add the principal:

   ```
   kadmin.local: addprinc -randkey HTTP/<host3>@EXAMPLE.COM
   ```

   Note

   This step will need to be performed each time the Spnego keytab is regenerated.

2. Use the following `kadmin.local` commands to add the HTTP Principal of each of the Ranger Admin and load-balancer nodes to the Spnego keytab file:

```
kadmin.local: ktadd -norandkey -kt /etc/security/keytabs/spnego.service.
keytab HTTP/ <host3>@EXAMPLE.COM
kadmin.local: ktadd -norandkey -kt /etc/security/keytabs/spnego.service.
keytab HTTP/ <host2>@EXAMPLE.COM
kadmin.local: ktadd -norandkey -kt /etc/security/keytabs/spnego.service.
keytab HTTP/ <host1>@EXAMPLE.COM
```

   Use the `exit` command to exit `ktadmin.local`.

3. Run the following command to check the Spnego keytab file:

```
klist -kt /etc/security/keytabs/spnego.service.keytab
```

   The output should include the principals of all of the nodes on which Ranger Admin and the load-balancer are installed. For example:

```
Keytab name: FILE:/etc/security/keytabs/spnego.service.keytab
KVNO Timestamp Principal ---- -----------------
-------------------------------------------------------
1 07/22/16 06:27:31 HTTP/ <host3>@EXAMPLE.COM
1 07/22/16 06:27:31 HTTP/ <host3>@EXAMPLE.COM
1 07/22/16 06:27:31 HTTP/ <host3>@EXAMPLE.COM
1 07/22/16 06:27:31 HTTP/ <host3>@EXAMPLE.COM
1 07/22/16 06:27:31 HTTP/ <host3>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:23 HTTP/ <host2>@EXAMPLE.COM
1 07/22/16 08:37:35 HTTP/ <host1>@EXAMPLE.COM
1 07/22/16 08:37:36 HTTP/ <host1>@EXAMPLE.COM 1
07/22/16 08:37:36 HTTP/ <host1>@EXAMPLE.COM 1 07/22/16
08:37:36 HTTP/ <host1>@EXAMPLE.COM 1 07/22/16 08:37:36
HTTP/ <host1>@EXAMPLE.COM 1 07/22/16 08:37:36 HTTP/
<host1>@EXAMPLE.COM
```

4. Use `scp` to copy the Spnego keytab file to every node in the cluster on which Ranger Admin and the load-balancer are installed. Verify that the `/etc/security/keytabs/spnego.service.keytab` file is present on all Ranger Admin and load-balancer hosts.

5. On the Ambari dashboard, select **Ranger > Configs > Advanced**, then select **Advanced ranger-admin-site**. Set the value of the ranger.spnego.kerberos.principal property to *.

6. Click **Save** to save the configuration, then restart Ranger Admin , only if you have completed all applicable post-upgrade tasks.

Ranger KMS

When upgrading Ranger KMS in an SSL environment to HDP-2.6, the properties listed in Step 2 under "Configuring the Ranger KMS Server" on the Enabling SSL for Ranger KMS will be moved to the "Advanced ranger-kms-site" section.

Druid Buffer Size Configuration

Problem:

After upgrading HDP to 2.6.3, the Druid Broker may fail with the following Out Of Memory Error:

```
Error in custom provider, java.lang.OutOfMemoryError
  at io.druid.guice.DruidProcessingModule.
getMergeBufferPool(DruidProcessingModule.java:124) (via modules: com.
google.inject.util.Modules$OverrideModule -> com.google.inject.util.Modules
$OverrideModule -> io.druid.guice.DruidProcessingModule)
  at io.druid.guice.DruidProcessingModule.
getMergeBufferPool(DruidProcessingModule.java:124) (via modules: com.
google.inject.util.Modules$OverrideModule -> com.google.inject.util.Modules
$OverrideModule -> io.druid.guice.DruidProcessingModule)
  while locating io.druid.collections.BlockingPool<java.nio.ByteBuffer>
 annotated with @io.druid.guice.annotations.Merging()
    for the 4th parameter of io.druid.query.groupby.strategy.
GroupByStrategyV2.<init>(GroupByStrategyV2.java:97)
  while locating io.druid.query.groupby.strategy.GroupByStrategyV2
    for the 3rd parameter of io.druid.query.groupby.strategy.
GroupByStrategySelector.<init>(GroupByStrategySelector.java:43)
  while locating io.druid.query.groupby.strategy.GroupByStrategySelector
    for the 1st parameter of io.druid.query.groupby.
GroupByQueryQueryToolChest.<init>(GroupByQueryQueryToolChest.java:104)
  at io.druid.guice.QueryToolChestModule.configure(QueryToolChestModule.
java:95) (via modules: com.google.inject.util.Modules$OverrideModule
 -> com.google.inject.util.Modules$OverrideModule -> io.druid.guice.
QueryRunnerFactoryModule)
  while locating io.druid.query.groupby.GroupByQueryQueryToolChest
  while locating io.druid.query.QueryToolChest annotated with @com.google.
inject.multibindings.Element(setName=,uniqueId=64, type=MAPBINDER, keyType=
java.lang.Class<? extends io.druid.query.Query>)
  at io.druid.guice.DruidBinders.queryToolChestBinder(DruidBinders.java:45)
 (via modules: com.google.inject.util.Modules$OverrideModule -> com.google.
inject.util.Modules$OverrideModule -> io.druid.guice.QueryRunnerFactoryModule
 -> com.google.inject.multibindings.MapBinder$RealMapBinder)
  while locating java.util.Map<java.lang.Class<? extends io.druid.query.
Query>, io.druid.query.QueryToolChest>
    for the 1st parameter of io.druid.query.MapQueryToolChestWarehouse.
<init>(MapQueryToolChestWarehouse.java:36)
  while locating io.druid.query.MapQueryToolChestWarehouse
  while locating io.druid.query.QueryToolChestWarehouse
    for the 1st parameter of io.druid.client.BrokerServerView.
<init>(BrokerServerView.java:91)
  at io.druid.cli.CliBroker$1.configure(CliBroker.java:95) (via modules: com.
google.inject.util.Modules$OverrideModule -> com.google.inject.util.Modules
$OverrideModule -> io.druid.cli.CliBroker$1)
  while locating io.druid.client.BrokerServerView
  at io.druid.cli.CliBroker$1.configure(CliBroker.java:96) (via modules: com.
google.inject.util.Modules$OverrideModule -> com.google.inject.util.Modules
$OverrideModule -> io.druid.cli.CliBroker$1)
  while locating io.druid.client.TimelineServerView
    for the 6th parameter of io.druid.server.BrokerQueryResource.
<init>(BrokerQueryResource.java:64)
```

```
  at io.druid.cli.CliBroker$1.configure(CliBroker.java:111) (via modules: com.
google.inject.util.Modules$OverrideModule -> com.google.inject.util.Modules
$OverrideModule -> io.druid.cli.CliBroker$1)
  while locating io.druid.server.BrokerQueryResource
Caused by: java.lang.OutOfMemoryError
        at sun.misc.Unsafe.allocateMemory(Native Method)
        at java.nio.DirectByteBuffer.<init>(DirectByteBuffer.java:127)
        at java.nio.ByteBuffer.allocateDirect(ByteBuffer.java:311)
        at io.druid.offheap.OffheapBufferGenerator.get(OffheapBufferGenerator.
java:53)
        at io.druid.offheap.OffheapBufferGenerator.get(OffheapBufferGenerator.
java:29)
        at io.druid.collections.DefaultBlockingPool.
<init>(DefaultBlockingPool.java:58)
        at io.druid.guice.DruidProcessingModule.
getMergeBufferPool(DruidProcessingModule.java:127)
        at io.druid.guice.DruidProcessingModule$$FastClassByGuice$$8e266e5c.
invoke(<generated>)
        at com.google.inject.internal.ProviderMethod$FastClassProviderMethod.
doProvision(ProviderMethod.java:264)
        at com.google.inject.internal.ProviderMethod$Factory.
provision(ProviderMethod.java:401)
        at com.google.inject.internal.ProviderMethod$Factory.
get(ProviderMethod.java:376)
        at com.google.inject.internal.ProviderToInternalFactoryAdapter$1.
call(ProviderToInternalFactoryAdapter.java:46)
```

Solution:

To work around this issue, manually reduce processing buffer size for both historical and broker:

```
druid.processing.buffer.sizeBytes= 134217728
```

More Information

Migrate Audit logs from DB to Solr

How to Regenerate Keytabs