Hortonworks Data Platform

Apache Ambari Upgrade

docs.cloudera.com

Hortonworks Data Platform: Apache Ambari Upgrade

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

We encourage our customers to deploy HDP 3.1 in production using one of two options:

1. Deploy a fresh HDP 3.1 cluster and migrate existing data using Data Lifecycle Manager or distributed copy (distcp).

2. Perform an in-place upgrade of an existing HDP 2.6.x cluster

Please reach out to your Hortonworks account team to participate in our Limited Upgrade Program. For in-place upgrades, we want to work with you very closely to ensure your cluster is ready and meets all the criteria to be successful.

As part of the planning exercise to upgrade to HDP 3.1, please consult the Hortonworks Support Matrix to ensure the Hortonworks products you are currently using are certified to work with HDP 3.1. Also, consult with your third-party software providers to ensure the products you're integrating with HDP are certified to work with HDP 3.1.

> ⚠️ **Important**
>
> Not all Management Packs are currently compatible with HDP 3.1. Hortonworks Support can help you assess the feasibility of your upgrade if you have Management Packs other than HDF installed.

Ambari and the stack managed by Ambari can be upgraded independently. Use the Ambari-2.7.3 Major Upgrade Guide to upgrade Ambari-2.6.2.2 to Ambari-2.7.3. Then, use Ambari-2.7.3 to upgrade HDP-2.6.x to HDP-3.1.0.

> ⚠️ **Important**
>
> **Ambari 2.7.3 only supports fully managing a HDP 3.1.x cluster**. Not all cluster management operations are supported when using Ambari 2.7.3 with HDP 2.6. Please see *Changes to Services and Views* for limitations.

Next Steps

Getting Ready to Upgrade Ambari and HDP [2]

More Information

Hortonworks Support Matrix

Changes to Services and Views [4]

Upgrade Ambari and the HDF Management Pack

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 ensure that you are using the most recent version of Ambari to perform your upgrade.

• 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, Ranger, Druid, Superset, and Oozie.

-

For Large Clusters

In a large cluster, NameNode startup processes can take a long time. NameNode startup time depends not only on host properties, but also on data volume and network parameters. To ensure that the Ambari requests to start the NameNode do not timeout during an upgrade, you should configure the Ambari NameNode restart timeout parameter, upgrade.parameter.nn-restart.timeout in `/etc/ambari-server/conf/ambari.properties` on the Ambari Server host. You may need to add the restart timeout parameter and value to the Ambari server host, following a default installation. For a large cluster, you should add ten percent to the usual time (in seconds) required to restart your NameNode. Although no standard way to determine an appropriate value exists, you may use the following guidance:

For example, record the time (seconds) required to restart the active NameNode for your current Ambari server version. If restarting takes 10 minutes, (600 seconds), then add

```
upgrade.parameter.nn-restart.timeout=660
```

to the `/etc/ambari-server/conf/ambari.properties` file on the Ambari Server host.

After adding or resetting the Ambari NameNode restart parameter, restart your Ambari server before starting the HDP upgrade.

```
ambari-server restart
```

-

For Ambari Upgrades

• This (Ambari 2.7.x) Upgrade Guide will help you upgrade your existing Ambari server to version 2.7.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-2.7.x release.

• Review supported Ambari Server database versions using the Hortonworks Support Matrix. Plan any necessary resources required for a database upgrade. You will upgrade your Ambari Server database during the Ambari Upgrade procedure.

• Ambari 2.7 **only** supports the following operations when running against a HDP 2.6 cluster:

  • Run Service Checks

  • Start, Stop, Restart a Service

  • Change Configuration

  • Enable & Disable Maintenance Mode

  • Disable Auto Start

  • Remove Services

  • Remove Components

  • Remove & Decommission Hosts

-

For HDP Cluster Upgrades

• Ensure sufficient disk space on `/usr/hdp/<version>` (roughly 3GB for each additional HDP release).

• Additional components will be added to the cluster as part of the HDP 3.0 upgrade including YARN ATSv2, YARN Registry DNS, and additional Hive Clients required for the Spark History Server. If your cluster has kerberos enabled, you must configure Ambari to manage the Kerberos admin credentials prior to the upgrade so the appropriate Kerberos principals can be created during the upgrade process.

- If your cluster includes Storm, document any running Storm topologies, as they will need to be stopped during the upgrade process.

Next Steps

Changes to Services and Views [4]

More Information

Managing Admin Credentials

Hortonworks Support Matrix

Using a Local Repository

Ambari 2.7.0 Release Notes

2.1. Changes to Services and Views

During the process of upgrading to Ambari 2.7 and HDP 3.0, additional components will be added to you cluster, and deprecated services and views will be removed.

Ambari 2.6 to Ambari 2.7

The Ambari 2.6 to Ambari 2.7 upgrade will remove the following views:

- Hive View 1.5, Hive View 2

- Hue To Ambari View Migration

- Slider

- Storm

- Tez

- Pig

The Ambari Pig View is deprecated in HDP 3.0 and later. Ambari does not enable Pig View. To enable Pig View in HDP 3.0 and later, you need to contact Hortonworks support for instructions that include how to install WebHCat using an Ambari management pack.

HDP 2.6 to HDP 3.0

The HDP 2.6 to HDP 3.0 upgrade will add the following components if YARN is deployed in the cluster being upgraded:

- YARN Registry DNS

- YARN Timeline Service V2.0 Reader

The HDP 2.6 to HDP 3.0 upgrade will remove the following services:

- Flume

- Mahout

- Falcon

- Spark 1.6

- Slider

These services will be removed automatically as part of the upgrade process, but they can be also be removed manually prior to the upgrade by following the steps below:

1. Log in to the Ambari UI.

2. Click on the service you wish to remove.

3. From the Service Actions menu, click Delete Service.

Next Steps

Upgrading the Cluster's Underlying OS [5]

2.2. Upgrading the Cluster's Underlying OS

With HDP 3.0 no longer supporting RHEL/CentOS/OEL 6, SLES 11, and Debian 7 all hosts in the cluster must be on a supported operating system before starting the upgrade from HDP 2.6 to HDP 3.0. For many, this is a process that will take time and orchestration between multiple teams within your organization. We've tried to outline two high-level guidelines for moving from one major operating system version to another:

In-Place & Restore: Perform an In-place OS refresh and use Ambari's Restore Host feature

Move & Decom: Move Masters and Decom/Recom Workers

Each option has pros and cons and some high-level decision criteria:

In-Place & Restore

This option should be used in medium to large clusters (25 or more nodes), with operational teams that have environment automation experience, and have followed best practices when setting up component High Availability and HDP directory structures (such as ensuring HDP component data and metadata is not stored on the root volume).

This option involves going through each host in the cluster and ensuring important metadata and data is stored on a volume that is not being used by the operating system, and leverages component high availability to maintain maximum cluster availability. When visiting each host, the host is shut down, the operating system volume is refreshed with the new version of the chosen operating system, the host is configured with the same IP address and hostname, all volumes are re-mounted, and the Ambari Agent is installed and configured. Once the host has re-joined the cluster, the Ambari Recover Host functionality is used to re-install, re-configure, and start services. Since the component data and metadata are stored on a volume that is not being used by the operating system, no data is lost and the host performs just like it did before, but with a new operating system version.

Move & Decom

This option should be used in smaller clusters (under 25 nodes), where operational teams may not have access to operating system and configuration management automation tooling or have not yet followed best practices when setting up HDP directory structures (such as ensuring HDP component data and metadata is not stored on the root volume).

This option involves decommissioning worker nodes and replacing them with worker nodes that have the new operating system version on them. For master nodes, the move master operation is used to move all masters off of a host, and on to a new host with the new operating system version on them. Decommissioning worker nodes can take a great deal of time, depending on the density of the nodes, and move master operations require many cluster services to be restarted, so this is a time-consuming process that requires multiple periods downtime, but it does not require any operating system level operations to be performed to accomplish.

Next Steps

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 Ambari [7]

- Upgrade Ambari [8]

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

Prepare	Stop Ambari	Upgrade Ambari Server & Agents	Upgrade the Ambari Server Schema	Start Ambari Server & Ambari Agents	Re-configure LDAP & JDBC
Perform the preparation steps, which include **making backups of critical cluster metadata**.	**Stop** the Ambari Server and on all hosts in the cluster stop the Ambari Agent	**Upgrade** the Ambari Server and on all hosts in the cluster upgrade the Ambari Agent	Upgrade the Ambari Server's **Database Schema**	**Start** the Ambari Server and on all hosts in the cluster start the Ambari Agent	Re-configure Ambari Server's LDAP integration and point Ambari to the right JDBC drivers

Next Steps

Preparing to Upgrade Ambari [7]

More Information

Hortonworks Support Matrix

3.1. Preparing to Upgrade Ambari

- Be sure to review the Ambari 2.7.0.0 release notes for Known Issues and Behavioral Changes.

- You **must** have root, administrative, or root-equivalent authorization on the Ambari Server host and all Ambari Agent hosts in the cluster.

- You **must** make a safe copy of the Ambari Server configuration file found at `/etc/ambari-server/conf/ambari.properties`.

- If your cluster is SSO-enabled, do not stop Knox before upgrading Ambari.

The following table lists recommended (✅), and unsupported (X) upgrade paths.

From / To	Ambari 2.7.x	Ambari 2.6.x
Ambari 2.6.x	✅	X
Ambari 2.5x	X	✅
Ambari 2.4x	X	✅

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 [8]

More Information

Upgrade SmartSense

Ambari 2.7.0.0 Release Notes

3.2. Upgrade Ambari

1. Turn off Auto Restart from Ambari Web by browsing to **Admin > Service Auto Start**. Set **Auto-Start Services** to **Disabled**. Click **Save**.

   From Ambari Web, browse to **Admin > Service Auto Start**. Set **Auto-Start Services** to **Disabled**. Click **Save**.

2. If you are running SmartSense in your cluster, stop the service.

   From **Ambari Web**, browse to **Services > SmartSense** and select **Stop** from the **Service Actions** menu. Then, select **Turn on Maintenance Mode** from the **Service Actions** menu.

3. If you are running Ambari Metrics in your cluster, stop the service and put it in Maintenance Mode.

   From **Ambari Web**, browse to **Services > Ambari Metrics** and select **Stop** from the **Service Actions** menu. Then, select **Turn on Maintenance Mode** from the **Service Actions** menu.

4. If you are running Log Search in your cluster, stop the service.

   From **Ambari Web**, browse to **Services > Log Search** and select **Stop** from the **Service Actions** menu. Then, select **Turn on Maintenance Mode** from the **Service Actions** menu.

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

```
ambari-server stop
```

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

```
ambari-agent stop
```

7. Back up your existing Ambari Server database.

   For example,to back up the default, embedded postgres db for Ambari:

   ```
    mkdir /root/backups/
   pg_dump -U ambari ambari > /root/backups/ambari-before-upgrade.sql
   ```

   The default password is: bigdata .

8. Upgrade your Ambari Server database to a current supported version, only if your Ambari Server database version is not supported on the target Ambari version.

   To verify current supported versions, see the Hortonworks Support Matrix.

   For example, if you are running postgres v9.4, you must upgrade postgres to v9.6 or v10.2. If you are running MariaDB 10.2, upgrading the Ambari Server database is not required.

   Consult the documentation for your database type and version for specific database upgrade instructions.

9. 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 from the following list:

   • **For RHEL/CentOS/Oracle Linux 7:**

   ```
   wget -nv https://username:password@archive.cloudera.com/p/ambari/2.x/2.7.
   0.0/centos7/ambari.repo -O /etc/yum.repos.d/ambari.repo
   ```

   • **For SLES 12:**

   ```
   wget -nv https://username:password@archive.cloudera.com/p/ambari/2.x/2.7.
   0.0/sles12/ambari.repo -O /etc/zypp/repos.d/ambari.repo
   ```

   • **For Ubuntu 14:**

   ```
   wget -nv https://username:password@archive.cloudera.com/p/ambari/2.x/2.7.
   0.0/ubuntu14/ambari.list -O /etc/apt/sources.list.d/ambari.list
   ```

   • **For Ubuntu 16:**

   ```
   wget -nv https://username:password@archive.cloudera.com/p/ambari/2.x/2.7.
   0.0/ubuntu16/ambari.list -O /etc/apt/sources.list.d/ambari.list
   ```

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

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

```
yum clean all
```

```
yum info ambari-server
```

In the info output, visually validate that there is an available version containing "2.7"

```
yum upgrade ambari-server
```

- **For SLES:**

```
zypper clean
```

```
zypper info ambari-server
```

In the info output, visually validate that there is an available version containing "2.7"

```
zypper up ambari-server
```

- **For Ubuntu/Debian:**

```
apt-get clean all
```

```
apt-get update
```

```
apt-cache show ambari-server | grep Version
```

In the info output, visually validate that there is an available version containing "2.7"

```
apt-get install ambari-server
```

> ### Important
>
> When performing upgrade on SLES, you will see a message "There is an update candidate for 'ambari-server', but it is from different vendor. Use 'zypper install ambari-server-2.7.0-750.noarch' to install this candidate". You will need to to use yast to update the package, as follows:
>
> a. From the command line run: > yast.
>
> ```
> > yast
> ```
>
> You will see command line UI for YaST program.
>
> b. Choose **Software > Software Management**, then click the **Enter** button.
>
> c. In the **Search Phrase** field, enter **ambari-server**, then click the **Enter** button.
>
> d. On the right side you will see the search result `ambari-server 2.7`. Click **Actions**, choose **Update**, then click the **Enter** button.
>
> e. Go to **Accept**, and click **enter**.

11.Check for upgrade success by noting progress during the Ambari Server installation process you started in Step 8.

- 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 0:2.7.0 Complete!
```

> **Important**
>
> 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-2.7.*.jar to `/tmp` before proceeding with upgrade.

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

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

```
yum upgrade ambari-agent
```

- **For SLES:**

```
zypper up ambari-agent
```

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

> **Important**
>
> When performing upgrade on SLES, you will see a message "There is an update candidate for 'ambari-agent', but it is from different vendor. Use 'zypper install ambari-agent-2.7.0-750.noarch' to install this candidate". You will need to to use yast to update the package, as follows:
>
> a. From the command line run: > yast
>
> ```
> > yast
> ```
>
> You will see command line UI for YaST program.
>
> b. Choose **Software > Software Management**, then click the **Enter** button.

11

       c. In the **Search Phrase** field, enter **ambari-agent**, then click the **Enter** button.

       d. On the right side you will see the search result `ambari-agent 2.7`. Click **Actions**, choose **Update**, then click the **Enter** button.

       e. Go to **Accept**, and click **enter**.

- **For Ubuntu/Debian:**

```
apt-get update
apt-get install ambari-agent
```

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

For RHEL/CentOS/Oracle Linux 7, Amazon Linux 2:	`rpm -qa	grep ambari-agent`
For SLES 12:	`rpm -qa	grep ambari-agent`
For Ubuntu 14:	`dpkg -l ambari-agent`
For Ubuntu 16:	`dpkg -l ambari-agent`
For Debian 9:	`dpkg -l ambari-agent`

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

```
ambari-server upgrade
```

When the Ambari Server database schema has been upgraded, you should see command output like this:

```
Ambari Server 'upgrade' completed successfully
```

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

```
ambari-server start
```

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

```
ambari-agent start
```

17.Open Ambari Web.

Point your browser to the Ambari Web UI:

- When Ambari Server is configured for HTTPS:

```
https://<your.ambari.server>:8443
```

- When Ambari Server is configured for HTTP:

```
http://<your.ambari.server>:8080
```

where <your.ambari.server> is the name of your ambari server host. For example, c7401.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.

18. 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".

> ⚠️ **Important**
>
> DO NOT RESTART these services unless future steps in the upgrade guide prompt you to do so. Manually restarting these services may significantly disrupt your upgrade. Ambari will restart each service automatically during the HDP upgrade.

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

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

20. If you have configured your cluster for Hive, Ranger 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.

> ⚠️ **Important**
>
> Ambari Metrics System, and SmartSense must be upgraded **after** first registering the HDP 3.0.x version and installing HDP 3.0.x packages. Please do not attempt to start Ambari Metrics or SmartSense until the HDP 3.0.x upgrade has completed.

Next Steps

Upgrading HDP [15]

More Information

Configuring Ambari Authentication for LDAP/AD

Using a new or existing database with Hive

4. Upgrading HDP

This topic describes available 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 2.7** 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.
>
> Ambari 2.7 does not support IOP-HDP migration. Customers migrating from IOP should use Ambari 2.6.1 to migrate IOP to HDP 2.6.4, then upgrade Ambari to 2.7 and use it to upgrade HDP 2.6.4 to HDP 3.0.

There is only one method for upgrading HDP-2.6 to HDP-3.0 with Ambari: **Express Upgrade**.

An **Express Upgrade** orchestrates the HDP upgrade in an order that will incur cluster downtime.

Next Steps

Prerequisites [16]

More Information

Preparing to Upgrade Ambari and HDP

4.1. Prerequisites

To perform an HDP upgrade using Ambari, your cluster must meet the following prerequisites. Meeting these prerequisites is essential for Ambari to know the cluster is in a healthy operating mode and can successfully manage the upgrade process.

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.
Hive Upgrade	The upgrade process does not back up the Hive MetaStore, nor does it compact ACID tables. Before upgrading Hive, you must: • Manually make a manual backup of your Hive metastore database after running the pre-upgrade tool, described later, and before upgrading. • If you have ACID tables in your Hive metastore, enable ACID operations using Ambari Web or set Hive configuration properties to enable ACID.
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, except for Ambari Metrics System, SmartSense, and Log Search.
Services Started	All Services must be started, except for Ambari Metrics System, SmartSense, and Log Search.

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.
KDC Admin Credentials	The Ambari Server will add new components as part of the HDP 2.6 to HDP 3.0 upgrade and needs to be configured to enable password encryption and saving the KDC admin credentials so necessary principals can be created. For steps on how to do this, see Encrypt Database and LDAP Passwords in Ambari and Managing Admin Credentials.
KDC Admin Host FQDN	Ensure that the KDC Admin host is set to a fully qualified domain name (FQDN) and not an IP address. You can check this setting by going to **Services > Kerberos > Configs > KDC Hosts**.
KDC kadmin principal	The MIT KDC integration in Ambari 2.7 has been improved to connect more securely with the Kerberos Administration server (kadmind). This increased security expects a Kerberos admin service principal to be present with a specifically formatted principal name. The format expected is `kadmin/ fully.qualified.kdc.hostname@REALM`. This expectation is a behavior change from previous versions of Ambari, where having such a Kerberos admin service principal was not required.
	This principal is present by default in most MIT KDC installations, but some customers have reported that this principal does not exist in their KDC. Due to this, it's recommended that before upgrading a kerberized cluster to Ambari 2.7, you ensure this principal exists in your KDC database.
All Prior Upgrades are Finalized	Any prior upgrade started with Ambari must be finalized.
Apache Zeppelin Notebooks and Configuration File Storage	In releases of Zeppelin earlier than HDP-2.6.3, notebooks and configuration files were stored on the local disk of the Zeppelin server. In HDP-2.6.3+, the default Zeppelin storage is HDFS.
	When upgrading to HDP-2.6.3+ from versions earlier than HDP-2.6.3, perform the steps described in Enabling HDFS Storage for Zeppelin Notebooks and Configuration in HDP-2.6.3+.

Next Steps

Prepare to Upgrade [18]

More Information

Register and Install Target Version [25]

Remove Service

4.2. Prepare to Upgrade

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

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

• Ambari database

• Hive Metastore database

• Oozie Server database

• Ranger database

• Ranger KMS database

• Druid database

• Superset Database

Next Steps

Checkpoint HDFS [18]

4.2.1. 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
```

Next Steps

Prepare Hive for upgrade [20]

4.2.2. Prepare Hive for upgrade

Before upgrading, contact your account team about your eligibility to upgrade to HDP 3.0.0. If you are eligible to upgrade, follow instructions to prepare Hive 2 in HDP 2.6.5 and later for upgrade to Hive 3. Upgrading Hive in releases earlier than HDP 2.6.5 is not supported.

Before you begin

- If not already installed, install JDK on the node running Hive Metastore.

- Check that the Hive Metastore is running. Connectivity between the tool and Hive MetaStore is mandatory.

- If you have ACID tables in your Hive metastore, enable ACID operations using Ambari Web or set these Hive configuration properties to enable ACID:

  - `hive.txn.manager=org.apache.hadoop.hive.ql.lockmgr.DbTxnManager`

  - `hive.support.concurrency=true`

  Failure to set these properties will result in corrupt or unreadable ACID tables.

- Optionally, shut down HiveServer2. Shutting down HiveServer2 is recommended, but not required, to prevent operations on ACID tables while the tool executes.

- The pre-upgrade tool might submit compaction jobs, so ensure that the cluster has sufficient capacity to execute those jobs. Set the `hive.compactor.worker.threads` property to accommodate your data.

- If you use Oracle as the backend database for Hive 1.x - Hive 3.x and the ojdbc7 JAR, replace this JAR with ojdbc6 JAR as described in the Cloudera Community "Unable to start Hive Metastore during HDP upgrade" article.

- Obtain permissions to perform the steps in preparing Hive for upgrade.

Required permissions

To perform some steps in this procedure, you need Hive service user permissions, or all the permissions to access Hive that Ranger provides. If you use Kerberos, you need to start Hive as the Hive service user with a valid ticket. The Hive service user is usually the default `hive` user. If you don't know who the Hive service user is, go to the Ambari Web UI, and click Cluster Admin > Service Accounts, and then look for Hive User.

To perform some steps in this procedure, you also need to login as the HDFS superuser. If you use Kerberos, you need to become the HDFS superuser with a valid ticket.

I. Backup Hive table data using a snapshot

Keep track of how many tables you have before upgrading for comparison after upgrading. Backup Hive table data as follows:

1. In Ambari, go to Services/Hive/Configs, and check the value of `hive.metastore.warehouse.dir` to determine the location of the Hive warehouse, /apps/hive/warehouse by default.

2. On any node in the cluster, as the HDFS superuser, enable snapshots. For example:

```
$ sudo su - hdfs

$ hdfs dfsadmin -allowSnapshot /apps/hive/warehouse
```

   Output is:

```
Allowing snaphot on /apps/hive/warehouse succeeded
```

3. Create a snapshot of the Hive warehouse. For example:

```
$ hdfs dfs -createSnapshot /apps/hive/warehouse
```

   Output includes the name and location of the snapshot:

```
Created snapshot /apps/hive/warehouse/.snapshot/
s20181204-164645.898
```

4. Start Hive as a user who has SELECT privileges on the tables. For example:

```
$ beeline

beeline> !connect jdbc:hive2://

Enter username for jdbc:hive2://: hive

Enter password for jdbc:hive2://: *********
```

   Output is, for example:

```
Connected to: Apache Hive (version 1.2.1000.2.6.5.0-292)

Driver: Hive JDBC (version 1.2.1000.2.6.5.0-292)
```

5. Identify all tables outside `/apps/hive/warehouse/`. For example:

```
hive> USE my_database;
```

```
hive> SHOW TABLES;
```

6. Determine the location of each table using the DESCRIBE command. For example:

```
hive> DESCRIBE FORMATTED my_table partition (dt='20181130);
```

7. Create a snapshot of the directory shown in the location section of the output.

8. Repeat steps 5-7 for each database and its tables outside `/apps/hive/warehouse/`.

II. For SparkSQL users only

Non-Acid, managed tables in ORC or in a Hive Native (but non-ORC) format that are owned by the POSIX user `hive` will not be SparkSQL-compatible after the upgrade unless you perform one of the following actions:

• Convert the tables to external Hive tables before the upgrade.

• Change the posix ownership to an owner other than `hive`.

You will need to convert managed, ACID v1 tables to external tables after the upgrade, as described later. The HDP 2.x and 3.x Table Type Comparison in section, "Hive Post-upgrade Tasks" identifies SparkSQL-incompatible table types.

III. Download the pre-upgrade tool JAR

1. SSH into the host running the Hive Metastore.

2. Change to the `/tmp` directory.

3. Execute the following command to download the pre-upgrade tool JAR:

```
$ wget http://repo.hortonworks.com/content/repositories/
releases/org/apache/hive/hive-pre-upgrade/3.1.0.3.0.0.0-1634/
hive-pre-upgrade-3.1.0.3.0.0.0-1634.jar
```

IV. Get a Kerberos ticket if you use Kerberos

If you use Kerberos, perform these steps; otherwise, skip these steps and go to the procedure for compacting Hive tables (no Kerberos).

1. Become the Hive service user. For example, run the following `su` command on Linux:

```
$ sudo su - hive
```

2. In a Kerberized cluster, run `kinit` to get a Kerberos ticket. For example:

```
$ kinit -kt /etc/security/keytabs/hive.service.keytab hive/
`hostname -f`
```

3. Set `-Djavax.security.auth.useSubjectCredsOnly=false` in a Kerberized environment if, after running `kinit`, you see the following error:

```
org.ietf.jgss.GSSException: No valid credentials provided
(Mechanism level: Failed to find any Kerberos tgt
```

4. Perform the procedure for compacting Hive tables below.

V. Optional: Override default table conversion

To override the default conversion of non-ACID tables to ACID (insert-only, managed table), change managed, non-ACID tables to external:

```
ALTER TABLE T3 SET TBLPROPERTIES ('EXTERNAL'='TRUE');
```

For more information about upgrade changes to tables, see HDP 2.x and 3.x Table Type Comparison.

VI. Run compaction on Hive tables

Using the downloaded JAR from step II and your Kerberos ticket (if you use Kerberos) from step III, perform the following procedure to run compaction on Hive tables.

1. Log in as the `hive` user.

   For example: `$ sudo su - hive`

2. Export the JAVA_HOME environment variable if necessary.

   For example: `$ export JAVA_HOME=[ path to your installed JDK ]`

3. Set STACK_VERSION to the HDP version you are running. For example:

   ```
   $ export STACK_VERSION=`hdp-select status hive-server2 | awk
   '{ print $3; }'`
   ```

4. Run the pre-upgrade tool command.

   ```
   $ $JAVA_HOME/bin/java -cp /usr/hdp/$STACK_VERSION/hive2/
   lib/derby-10.10.2.0.jar:/usr/hdp/$STACK_VERSION/hive2/lib/
   *:/usr/hdp/$STACK_VERSION/hadoop/*:/usr/hdp/$STACK_VERSION/
   hadoop/lib/*:/usr/hdp/$STACK_VERSION/hadoop-mapreduce/
   *:/usr/hdp/$STACK_VERSION/hadoop-mapreduce/lib/*:/usr/
   hdp/$STACK_VERSION/hadoop-hdfs/*:/usr/hdp/$STACK_VERSION/
   hadoop-hdfs/lib/*:/usr/hdp/$STACK_VERSION/hadoop/etc/
   hadoop/*:/tmp/hive-pre-upgrade-3.1.0.3.0.0.0-1634.jar:/
   usr/hdp/$STACK_VERSION/hive/conf/conf.server
   org.apache.hadoop.hive.upgrade.acid.PreUpgradeTool>
   {hive_log_dir}/pre_upgrade_{target_version}.log
   ```

   The output indicates whether you need to perform compaction or not:

   • In the `/tmp` directory, scripts named `compacts_nnnnnnnnnnnnn.sql` appear that
     contain ALTER statements for compacting tables. For example:

     ```
     ALTER TABLE default.t COMPACT 'major';
     ```

     ```
     - Generated total of 1 compaction commands
     ```

```
- The total volume of data to be compacted is 0.001155MB
```

From the volume of data to be compacted, you can gauge how long the actual upgrade might take.

- If no scripts appear, a message in the output says you do not need to compact tables:

```
... org.apache.hadoop.hive.upgrade.acid.PreUpgradeTool - No
compaction is necessary
```

For more information about the pre-upgrade tool command, see the Pre-upgrade Tool Command Reference below.

5. Check the following logs on the Hive Metastore host for any errors:

- `{hive_log_dir}/pre_upgrade_{target_version}.log`

- `/tmp/hive/hive.log`

If there are no errors, go to the next step; otherwise, resolve the errors, and repeat this procedure.

6. On the node where the Hive Metastore resides, log in as a user who has privileges to alter the Hive database.

7. Start Beeline as the Hive service user. For example:

```
$ beeline -u 'jdbc:hive2://<Metastore host name>:10000' -n hive
```

8. On the Hive command line run the compaction script. For example:

```
hive> !run /tmp/compacts_nnnnnnnnnnnnn.sql
```

Output confirms that compaction is queued:

```
INFO : Compaction enqueued with id 3
```

…

9. Proceed to back up the Hive Metastore. This is a mandatory step.

VII. Back up Hive Metastore

After compaction, immediately before upgrading, backup Hive Metastore as follows:

> **⚠ Important**
>
> **Making a backup is critical to prevent data loss.**

1. On the node where the database you use for Hive Metastore resides, back up Hive Metastore before upgrading to HDP. For example, in MySQL, dump each database as follows:

```
mysqldump <hive_db_schema_name> > </path/to/dump_file>
```

If you use another database for the Hive Metastore, use the equivalent command, such as `export` for Postgres, to dump the database.

2. Proceed to upgrade HDP, assuming no Hive update, delete, or merge occurred after compaction; otherwise, repeat the compaction and Hive Metastore backup procedures, and then upgrade HDP.

Pre-upgrade tool command reference

You can use the following key options with the pre-upgrade tool command:

- -execute

  Use this option only when you want to run the pre-upgrade tool command in Ambari instead of on the Beeline command line. Using Beeline is recommended. This option automatically executes the equivalent of the generated commands.

- -location

  Use this option to specify the location to write the scripts generated by the pre-upgrade tool.

You can append `--help` to the command to see all command options. For example:

```
$ cd <location of downloaded pre-upgrade tool>

$ $JAVA_HOME/bin/java -
Djavax.security.auth.useSubjectCredsOnly=false -cp /usr/
hdp/$STACK_VERSION/hive2/lib/derby-10.10.2.0.jar:/usr/
hdp/$STACK_VERSION/hive2/lib/*:/usr/hdp/$STACK_VERSION/
hadoop/*:/usr/hdp/$STACK_VERSION/hadoop/lib/*:/usr/hdp/
$STACK_VERSION/hadoop-mapreduce/*:/usr/hdp/$STACK_VERSION/
hadoop-mapreduce/lib/*:/usr/hdp/$STACK_VERSION/hadoop-
hdfs/*:/usr/hdp/$STACK_VERSION/hadoop-hdfs/lib/*:/usr/
hdp/$STACK_VERSION/hadoop/etc/hadoop/*:/tmp/hive-pre-
upgrade-3.1.0.3.0.0.0-1634.jar:/usr/hdp/$STACK_VERSION/hive/conf/
conf.server org.apache.hadoop.hive.upgrade.acid.PreUpgradeTool --
help
```

In a Kerberized environment, if you see the errors after running `kinit`, include the following option when you run the pre-upgrade tool command, as shown in the –help example above:

```
-Djavax.security.auth.useSubjectCredsOnly=false
```

Next Steps

4.3. Register and Install Target Version

Before you use Ambari to perform the stack upgrade, you must register the software repositories for the new target version with Ambari and then install the software on all hosts in the cluster.

Register Target Version

Steps

1. Log in to Ambari.

2. Browse to **Cluster 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 2.6.5.0 release, you would see something like **HDP-2.6.5.0-292** 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**.

      Available HDP minor versions display on tabs. When you click a tab, Ambari displays available maintenance versions for that HDP Stack on a drop-down list. When you click a specific maintenance version, a list of available Services and their version displays in a table.

   b. **Choose HDP Version**.

      If your Ambari host has internet access, available maintenance versions display as options in a drop-down list. If you have a Version Definition File (VDF) 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**.

      • Use private Repository

      Using a private repository requires internet connectivity. To use the private software repositories, see the list of available HDP Repositories for each OS.

      • Use Local Repository

      Using a local repository requires that you have configured the software in a repository available in your network. If you are using a local repository, enter the Base URLs for your local repository.

7. Click **Save**.

8. Click **Dashboard**.

Install Target Version

Steps

1. Browse to **Cluster Admin > Stack and Versions**.

2. Click the **Versions** tab.

3. On a registered target version, click **Install Packages** 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.

   When the installation completes, the **Upgrade** button replaces the **Install Packages** button.

Next Steps

Upgrading Ambari Metrics System and SmartSense [27]

More Information

Managing Versions

Using a Local Repository

4.4. Upgrading Ambari Metrics System and SmartSense

As part of the Ambari 2.7 release, the schema for the Ambari Metrics System has significantly changed. This change requires additional steps to be performed before upgrading to HDP 3.0. As SmartSense depends on the Ambari Metrics System, SmartSense also requires similar handling. The following steps will guide you through upgrading the Ambari Metrics System and SmartSense.

> **Note**
>
> Before following these steps, ensure that the Ambari Metrics System and SmartSense are **stopped** and in **Maintenance Mode**. The HDP 3.0.0.0 version should be registered and the packages should be installed.

Next Steps

Upgrading Ambari Metrics [28]

More Information

Upgrade Ambari

Register and Install Target Version [25]

4.4.1. Upgrading Ambari Metrics

Steps

1. Confirm that Ambari Metrics service is stopped and in **Maintenance Mode**.

   a. If Ambari Metrics service is not 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:

   For RHEL/CentOS/Oracle Linux:

   ```
   yum clean all
   ```

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

   For SLES:

   ```
   zypper clean
   ```

   ```
   zypper up ambari-metrics-monitor ambari-metrics-hadoop-sink
   ```

   For Ubuntu/Debian:

   ```
   apt-get clean all
   ```

   ```
   apt-get update
   ```

   ```
   apt-get install ambari-metrics-assembly
   ```

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

   For RHEL/CentOS/Oracle Linux:

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

   For SLES:

   ```
   zypper up ambari-metrics-collector
   ```

   For Ubuntu/Debian:

   ```
   apt-get clean all
   ```

   ```
   apt-get update
   ```

   ```
   apt-get install ambari-metrics-collector
   ```

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

   For RHEL/CentOS/Oracle Linux:

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

   For SLES:

   ```
   zypper up ambari-metrics-grafana
   ```

For Ubuntu/Debian:

```
apt-get clean all
```

```
apt-get update
```

```
apt-get install ambari-metrics-grafana
```

> ⚠️ **Important**
>
> **DO NOT START the Ambari Metrics System** service. It will be started automatically during the HDP upgrade process.

Next Steps

Upgrading SmartSense [29]

4.4.2. Upgrading SmartSense

1. Confirm that SmartSense and Ambari Metrics are stopped and in **Maintenance Mode**.

   a. If SmartSense is not stopped, from **Ambari Web**, browse to **Services > SmartSense** and select **Stop** from the **Service Actions** menu.

   b. If Ambari Metrics is not in **Maintenance Mode**, From **Ambari Web**, browse to **Services > Ambari Metrics** and select **Turn On Maintenance Mode** from the **Service Actions** menu.

2. Upgrade binaries on the HST server and all HST agents on *every node* in the cluster, assuming that the Ambari repository is configured on all nodes in the cluster:

   • RHEL or CentOS

   ```
   yum clean all
   yum info smartsense-hst
   ```

   In the `info` output, visually validate that there is an available version containing "1.5.x":

   ```
   yum upgrade smartsense-hst
   ```

   • SLES

   ```
   zypper clean
   zypper info smartsense-hst
   ```

   In the `info` output, visually validate that there is an available version containing "1.5.x":

   ```
   zypper up smartsense-hst
   ```

   • Ubuntu or Debian

   ```
   apt-get clean all
   apt-get update
   apt-cache show smartsense-hst | grep Version
   ```

In the info output, visually validate that there is an available version containing "1.5.x":

```
apt-get install smartsense-hst
```

3. Upgrade Ambari service and Ambari view by running the **hst upgrade-ambari-service** command as the root user from the machine running the Ambari server. You can run the command in the interactive or non-interactive mode:

Interactive mode example:

```
# hst upgrade-ambari-service
Please enter Ambari Server hostname (ambari-server.hortonworks.local):
Please enter Ambari Server port (8080):
Please enter Ambari admin user id (admin):
Please enter password for admin:

Un-installing old view ...
Installing new view ...
Removing deprecated alerts ...
Updating SmartSense configurations in Ambari ...

SmartSense service upgrade completed!
NOTE: It is required to restart Ambari Server for changes to reflect. Please
 restart ambari using 'ambari-server restart'
```

Non-interactive mode example:

```
# hst upgrade-ambari-service -u admin -p 8080 -H ambari-server.hortonworks.
local -P MySecurePassword123
Un-installing old view ...
Installing new view ...
Removing deprecated alerts ...
Updating SmartSense configurations in Ambari ...
SmartSense service upgrade completed!
NOTE: It is required to restart Ambari Server for changes to reflect. Please
 restart ambari using 'ambari-server restart'
```

4. Restart the Ambari server:

```
# ambari-server restart
```

5. If you have HST Gateway installed, you need to also upgrade your HST Gateway:

   • If the HST Gateway is installed on the same node as HST Server or HST Agent, then the HST Gateway will get upgraded along with them.

   • If the HST Gateway is a standalone node outside of the cluster, perform upgrade steps described in Upgrading SmartSense Gateway.

   ⚠️ **Important**

   It is very important to make sure you **DO NOT START the SmartSense** service. It will be started automatically during the HDP upgrade process.

Next Steps

Backing up Ambari Infra Data & Migrating Atlas Data [31]

4.5. Backing up Ambari Infra Data & Migrating Atlas Data

In Ambari 2.7, Ambari Infra Solr uses Solr 7. In order for data stored in the existing Solr 5 implementation used by Ambari 2.6, the data stored in Ambari Infra must be backed up and the Ambari Infra Solr services must be upgraded. The next section will walk you through steps to both back up the data stored in Ambari Infra, upgrade the Ambari Infra components. The backed up data will be migrated and restored after the HDP upgrade is complete.

In HDP 3.0 Atlas data must be migrated from one graph implementation to another. This requires backing up the Atlas graph data before the HDP upgrade, so that it can be automatically migrated during the HDP upgrade process. The next sections will outline the steps required to backup and configure Atlas for this migration process.

Next Steps

Back up and Upgrade Ambari Infra [31]

4.5.1. Back up and Upgrade Ambari Infra

The Ambari Infra Solr instance is used to index data for Atlas, Ranger, and Log Search. The version of Solr used by Ambari Infra in Ambari 2.6 is Solr 5. The version of Solr used by the Ambari Infra in Ambari 2.7 is Solr 7. When moving from Solr 5 to Solr 7 indexed data needs to be backed up from Solr 5, migrated, and restored into Solr 7 as there are on disk format changes, and collection-specific schema changes. The Ambari Infra Solr components must also be upgraded. Fortunately scripts are available to do both, and are explained below.

This process will be broken up into four steps:

Generate Migration Config	The migration utility requires some basic information about your cluster and this step will generate a configuration file that captures that information.
Back up Ambari Infra Solr Data	This process will backup all indexed data either to a node-local disk, shared disk (NFS mount), or HDFS filesystem.
Remove existing collections & Upgrade Binaries	This step will remove the Solr 5 collections, upgrade Ambari Infra to Solr 7, and create the new collections with the upgraded schema required by HDP 3.0 services. This step will also upgrade LogSearch binaries if they are installed.
Migrate & Restore	This step will migrate the backed up data to the new format required by Solr 7 and restore the data into the new collections. **This step will be completed after the HDP 3.0 Upgrade has been completed in the Post-upgrade Steps section of the upgrade guide**

Generate Migration Config

The utility used in this process is included in the ambari-infra-solr-client package. This package must be upgraded before the utility can be run. To do this:

1. SSH into a host that has a Infra Solr Instance installed on it. You can locate this host by going to the Ambari Web UI and clicking **Hosts**. Click on the Filter icon and type `Infra Solr Instance: All` to find each host that has an Infra Solr Instance installed on it.

2. Upgrade the ambari-infra-solr-client package.

```
yum clean all
```

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

3. If you are using a custom username for running Infra Solr, for example a username that is not 'infra-solr' additional scripts need to be downloaded. To do this, again only if you are using a custom username for Infra Solr, perform the following steps:

   a. ```
   wget --no-check-certificate -O
   /usr/lib/ambari-infra-solr-client/migrationConfigGenerator.py
   https://raw.githubusercontent.com/apache/ambari/release-2.7.3/ambari-
   infra/ambari-infra-
   solr-client/src/main/python/migrationConfigGenerator.py
   ```

   b. ```
   chmod +x /usr/lib/ambari-infra-solr-client/migrationConfigGenerator.py
   ```

   c. ```
   wget --no-check-certificate -O /usr/lib/ambari-infra-solr-client/
   migrationHelper.py
   https://raw.githubusercontent.com/apache/ambari/release-2.7.3/ambari-
   infra/ambari-infra-
   solr-client/src/main/python/migrationHelper.py
   ```

   d. ```
   chmod +x /usr/lib/ambari-infra-solr-client/migrationHelper.py
   ```

4. You can now proceed to configuring and running the migration tool from the same host.

   Run the following commands as root, or with a user that has sudo access:

   Export the variable that will hold the full path and filename of the configuration file.

   ```
   export CONFIG_INI_LOCATION=ambari_solr_migration.ini
   ```

   Ensure the script generates cleanly and there are no yellow warning texts visible. If so, review the yellow warnings.

5. Run the `migrationConfigGenerator.py` script, located in the `/usr/lib/ambari-infra-solr-client/` directory, with the following parameters:

–ini-file $CONFIG_INI_LOCATION	This is the previously exported environmental variable that holds the path and filename of the configuration file that will be generated.
–host ambari.hortonworks.local	This should be the hostname of the Ambari Server.
–port 8080	This is the port of the Ambari Server. If the Ambari Server is configured to use HTTPS, please use the

	HTTPS port and add the `-s` parameter to configure HTTPS as the communication protocol.
–cluster cl1	This is the name of the cluster that is being managed by Ambari. To find the name of your cluster, look in the upper right and corner of the Ambari Web UI, just to the left of the background operations and alerts.
–username admin	This is the name of a user that is an "Ambari Admin" .
–password admin	This is the password of the aforementioned user.
–backup-base-path=/my/path	This is the location where the backed up data will be stored. Data will be backed up to this local directory path on each host that is running an Infra Solr instance in the cluster. So, if you have 3 Infra Solr server instances and you use –backup-base-path=/home/solr/backup, this directory will be created on all 3 hosts and the data for that host will be backed up to this path. If you are using a shared file system that is mounted on each Infra Solr instance in the cluster, please use the `--shared-drive` parameter instead of `--backup-base-path`. The value of this parameter should be the path to the mounted drive that will be used for the backup. When this option is chosen, a directory will be created in this path for each Ambari Infra Solr instance with the backed up data. For example, if you had an NFS mount /export/solr on each host, you would use `--shared-drive=/exports/solr`. Only use this option if this path exists and is shared amongst all hosts that are running the Ambari Infra Solr.
–java-home /usr/jdk64/jdk1.8.0_112	This should point to a valid Java 1.8 JDK that is available at the same path on each host in the cluster that is running an Ambari Infra Solr instance.
If the Ranger Audit collection is being stored in HDFS, please add the following parameter, –ranger-hdfs-base-path	The value of this parameter should be set to the path in HDFS where the Solr collection for the Ranger Audit data has been configured to store its data. **Example:** `--ranger-hdfs-base-path=/user/infra-solr`

Example Invocations:

If using HTTPS for the Ambari Server:

```
/usr/bin/python /usr/lib/ambari-infra-solr-client/
migrationConfigGenerator.py --ini-file $CONFIG_INI_LOCATION --
```

```
host c7401.ambari.apache.org --port 8443 -s --cluster cl1 --
username admin --password admin --backup-base-path=/my/path --
java-home /usr/jdk64/jdk1.8.0_112
```

If using HTTP for the Ambari Server:

```
/usr/bin/python /usr/lib/ambari-infra-solr-client/
migrationConfigGenerator.py --ini-file $CONFIG_INI_LOCATION
--host c7401.ambari.apache.org --port 8080 --cluster cl1 --
username admin --password admin --backup-base-path=/my/path --
java-home /usr/jdk64/jdk1.8.0_112
```

Back up Ambari Infra Solr Data

Once the configuration file has been generated, it's recommended to review the ini file created by the process. There is a configuration section for each collection that was detected. If, for whatever reason, you do not want to backup a specific collection you can set enabled = false and the collection will not be backed up. Ensure that enabled = true is set for all of the collections you do wish to back up. Only the Atlas, and Ranger collections will be backed up. Log Search will not be backed up.

To execute the backup, run the following command from the same host on which you generated the configuration file:

```
# /usr/lib/ambari-infra-solr-client/ambariSolrMigration.sh --ini-
file $CONFIG_INI_LOCATION --mode backup | tee backup_output.txt
```

During this process, the script will generate Ambari tasks that are visible in the Background Operations dialog in the Ambari Server.

Once the process has completed, please retain the output of the script for your records. This output will be helpful when debugging any issues that may occur during the migration process, and the output contains information regarding the number of documents and size of each backed up collection.

Remove Existing Collections & Upgrade Binaries

Once the data base been backed up, the old collections need to be deleted, and the Ambari Infra Solr, and Log Search (if installed) components need to be upgraded. To do all of that, run the following script:

```
# /usr/lib/ambari-infra-solr-client/ambariSolrMigration.sh --ini-
file $CONFIG_INI_LOCATION --mode delete | tee delete_output.txt
```

During this process, the script will generate Ambari tasks that are visible in the Background Operations dialog in the Ambari Server.

Once the process has completed, please retain the output of the script for your records. This output will be helpful when debugging any issues that may occur during the migration process.

Troubleshooting

What to do if Solr restarts immediately, before you upgrade the Solr instance packages:

Next Steps

4.5.2. Migrating Atlas Data

In HDP 3.0, Atlas uses the JanusGraph instead of Titan for data storage and processing. As part of the HDP 3.0 upgrade, data must be migrated from Titan to JanusGraph. Perform the following steps to migrate the Atlas metadata from Titan to JanusGraph.

Before upgrading HDP, use one of the following methods to determine the size of the Atlas metadata:

• Click **SEARCH** on the Atlas web UI, then slide the green toggle button from **Basic** to **Advanced**. Enter the following query in the **Search by Query** box, then click **Search**.

```
Asset select count()
```

• Run the following Atlas metrics REST API query:

```
curl -g -X GET -u admin:admin -H "Content-Type: application/json"
-H"Cache-Control: no-cache"
"http://<atlas_server>:21000/api/atlas/admin/metrics"
```

Either of these methods returns the number of Atlas entities, which can be used to estimate the time required to export and import the Atlas metadata. This time varies depending on the cluster configuration. The following estimates are for a node with a 4 GB RAM quad-core processor with both the Atlas and Solr servers on the same node:

• Estimated duration for export: 2 million entities per hour.

• Estimated duration for import: 0.75 million entities per hour.

1. In the Ambari Web UI, click **Atlas**, then select **Actions > Stop**.

2. Make sure that HBase is Running. If it is not, in the Ambari Web UI, click **HBase**, then select **Actions** > **Start**.

3. SSH to the host where your Atlas Metadata Server is running.

4. Start exporting Atlas metadata, using the following command format:

```
python /usr/hdp/3.0.0.0-<build_number>/atlas/tools/migration-
exporter/atlas_migration_export.py -d <output directory>
```

For example, export the metadata to an output directory called /atlas_metadata.

While running, the Atlas migration tool prevents Atlas use, and blocks all REST APIs and Atlas hook notification processing. As described previously, the time it takes to export the Atlas metadata depends on the number of entities and your cluster configuration. You can use the following command to display the export status:

```
tail -f /var/log/atlas/atlas-migration-exporter.log
```

When the export is complete, a file named atlas-migration-data.json is created in the output directory specified using the -d parameter. This file contains the exported Atlas entity data.

The HDP upgrade starts Atlas automatically, which initiates the migration of the uploaded HDP-2.x Atlas metadata into HDP-3.x. During the migration import process, Atlas blocks all REST API calls and Atlas hook notification processing. In order for this migration process to succeed, Atlas must be configured with the location of the exported data. Use the following steps to configure the atlas.migration.data.filename property.

1. In Ambari, select **Services > Atlas > Configs > Advanced > Custom application-properties**.

2. Click **Add Property**, and add the `atlas.migration.data.filename` property. Set the value to the location of the directory containing your exported Atlas metadata.
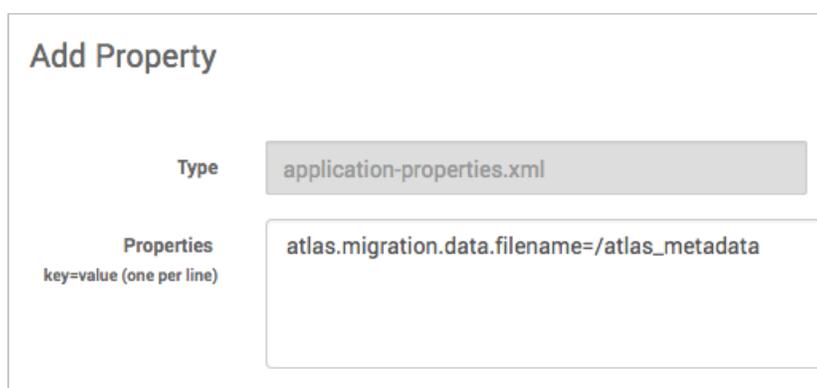
For example:

3. **Save** the configuration.

4. Click **Services** > **Atlas** > **Restart** > **Restart All Affected**.

5. Since the configuration has been changed, you need to re-run the Atlas service check by clicking **Actions** > **Run Service Check.**

> **Note**
>
> During the HDP upgrade, you can use the following Atlas API URL to display the migration status:
>
> `http://[atlas_server]:21000/api/atlas/admin/status`
>
> The migration status is displayed in the browser window:
>
> `{"Status":"Migration","currentIndex":139,"percent":67,"startTimeUTC":'`

Next Steps

4.6. Perform the Upgrade

To upgrade your HDP version:

Steps

1. Log in to Ambari.

2. Browse to **Cluster Admin > Stack and Versions**.

3. Click the **Versions** tab.

   The registered and installed target HDP version displays an **Upgrade** button.

4. Click **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 the **Express Upgrade** method. **(Only supported method for upgrading from HDP 2.6 to 3.0)**

   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**.

Next Steps

Perform Express Upgrade [37]

More Information

Register and Install Target Version [25]

Prerequisites [16]

4.6.1. 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 apps on YARN queues, and deactivate & kill all running Storm topologies. If you have any applications running on Slider, it is recommended to stop them, since Slider would be removed as part of the Upgrade process.	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 Respository	Updates the stack version in Ambari to the target version.	None
Add New YARN Components	In HDP 3.0, there are two new YARN components: YARN Registry DNS, and Timeline Reader. Both of these components are automatically added to the cluster.	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).	In an SSO-enabled cluster: If the cookie is lost/session is expired:client should use local login to access the Ambari and proceed further. For example: <ambari_host:ambari_port>/#/login/local
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, Hive, Spark2 and others.	None
Set Version on All Hosts	Sets the HDP version on all hosts to the target HDP version.	None

Stage	Description	Action Required
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.	When prompted to remove hbase snapshots, look for snapshots matching $TABLE-ru-$TIMESTAMP Prompted to Finalize or Finalize Later or Downgrade.

4. When the upgrade stages complete, you may choose to **Finalize** the upgrade, or to **Finalize Later**. Finalizing later gives you a chance to perform more validation on the cluster.

> ### 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.

Next Steps

Post-upgrade Tasks [39]

More Information

Register and Install Target Version [25]

Prerequisites [16]

4.7. Post-upgrade Tasks

Atlas Migration and HBase Hook Settings

- The settings for the Atlas migration need to be removed once the upgrade has been completed. On the Ambari Dashboard, select **Atlas > Configs > Advanced > Custom application-properties**, then click the red Remove button to remove the `atlas.migration.data.filename` property. Restart **Atlas** when prompted by Ambari.

- When upgrading to HDP-3.0 from earlier versions, you may need to manually enable the HBase hook. On the Ambari dashboard, select **HBase > Configs > Advanced hbase-env**, then select the **Enable Atlas Hook** checkbox. Click **Save**, then restart HBase and any other services that require a restart.

Ambari Metrics, SmartSense, LogSearch

Take the Ambari Metrics, SmartSense, and Log Search Services out of Maintenance Mode by choosing **Actions > Turn Off Maintenance Mode** from each Service page.

Ambari Infra-Migrate & Restore

Follow the steps below to restore the data previously backed up:

1. SSH to the host where the `migrationConfigGenerator.py` was run prior to the HDP Upgrade. This will be from one of your Ambari Infra Solr instances. Please ensure you are in the current working directory containing the `ambari_solr_migration.ini` file.

2. Export the variable used to hold the path to the ini file.

   ```
   export CONFIG_INI_LOCATION=ambari_solr_migration.ini
   ```

3. Migrate the data to ensure it's in the right format to import into Solr 7. Please note that this script can take a long time to run depending on the volume of data backed up. It is recommended to run this script using the nohup command.

   ```
   nohup /usr/lib/ambari-infra-solr-client/ambariSolrMigration.sh
   --ini-file $CONFIG_INI_LOCATION --mode migrate-restore >
   migrate_restore_output.txt 2>&1 &
   ```

   To observe the progress of the migrate and restore process, simply tail the migrate_restore_output.txt file.

   ```
   tail -f migrate_restore_output.txt
   ```

4. Re-index the migrated data into your current collections so the backed up data is visible in all of the tools using the Infra Solr instances. Please note that this script can take a long time to run depending on the volume of data backed up. It is recommended to run this script using the nohup command.

   ```
   nohup /usr/lib/ambari-infra-solr-client/ambariSolrMigration.sh
   --ini-file $CONFIG_INI_LOCATION --mode transport >
   transport_output.txt 2>&1 &
   ```

   To observe the progress of the transport process, simply tail the transport_output.txt file:

   ```
   tail -f transport_output.txt
   ```

Migrate Ambari Metrics Data

Use the following steps to migrate data from the previous AMS schema to the new AMS schema.

1. Ensure the Ambari Metrics System is started. If it is not, in the Ambari Web UI, click **Ambari Metrics**, then select **Actions** > **Start**.

2. SSH into a host that is running a Metrics Collector. You can locate this host by going to the Ambari Web UI and clicking **Hosts**. Click on the Filter icon and type in "Metrics Collector: All" to find each host that has a Metrics Collector installed on it.

3. SU to the Ambari Metrics user. This is 'ams' by default, but if you don't know which user is configured in your cluster go to the Ambari Web UI and click **Cluster Admin** > **Service Accounts**, and then look for "Ambari Metrics User".

```
# su ams
```

4. Run the command to migrate data from the old Ambari Metrics schema to the new.

```
$ /usr/sbin/ambari-metrics-collector --config /etc/ambari-
metrics-collector/conf/ upgrade_start /etc/ambari-metrics-
collector/conf/metrics_whitelist
```

> **Note**
>
> The default time period for data migration is one month (2592000000 milli seconds), which can be overwritten if required. For overwriting the data migration to more than one month, you must provide the timeframe upto which data need to be migrated along with the command.
>
> ```
> /usr/sbin/ambari-metrics-collector --config /etc/ambari-
> metrics-collector/conf/ upgrade_start /etc/ambari-
> metrics-collector/conf/metrics_whitelist "31556952000"
> ```
>
> "31556952000" is the amount of days in milliseconds to be preserved and 31556952000 stands for 1 year.

5. Once the upgrade process has started, logs are available in the <ams-log-dir>/ambari-metrics-migration.log file.

Update Ranger Passwords

Ranger password validation has been updated in HDP 3.0, and to conform to these new password policies, the following Ranger passwords need to be updated to ensure that they have at least 8 characters with minimum one alphabet and one numeric. These passwords cannot contain the following special characters: " ' \ `

• Ranger Admin

The following new passwords need to be populated with valid passwords that also conform to the password policy:

• Ranger Usersync User's Password

• Ranger Tagsync User's Password

• Ranger KMS Keyadmin User's Password

If Applicable: Configure Custom Spark SQL Warehouse Directory

Since Spark 2.0.0, Spark references `spark.sql.warehouse.dir` as the default Spark SQL Warehouse location. In HDP-2.6.x, the standard value for the `spark.sql.warehouse.dir` property is `/apps/hive/warehouse`. On an Ambari cluster with HDP-2.6.x, this property must be set manually both in "Custom spark2-defaults" and "Custom spark2-thrift-sparkconf".

When upgrading from HDP-2.6.x to HDP-3.x, Ambari migrates the `spark.sql.warehouse.dir` property to "Advanced spark2-defaults" and "Advanced spark2-thrift-sparkconf", and changes the value of this property to `/apps/spark/`

`warehouse`. This is done to accommodate the separate Spark and Hive catalogs introduced in HDP-3.x.

If you used a custom setting for the `spark.sql.warehouse.dir` property in HDP-2.6.x, the Ambari upgrade to HDP-3.x ignores the custom setting and sets the value of the `spark.sql.warehouse.dir` property to `/apps/spark/warehouse` in both "Advanced spark2-defaults" and "Advanced spark2-thrift-sparkconf".

If you want to use a custom Spark SQL warehouse after upgrading to HDP-3.x, select **Spark2 > Configs**, then use **Add Property** in "Advanced spark2-defaults" and "Advanced spark2-thrift-sparkconf" to update the value of the `spark.sql.warehouse.dir` property with the custom setting.

4.8. Hive Post-upgrade Tasks

You might need to perform the following tasks after upgrading to HDP 3.x.

• Changing file locations

• Adding backticks to table references using dot notation

• Updating HDFS paths in Ranger policies

• Using the Hive Warehouse Connector to access Spark data

• Installing DAS in lieu of Hive View and Tez View

Changes include the location of tables, the HDFS path to the Hive warehouse, file ownership, table types, and ACID-compliance. You need to understand the following changes that occur during the upgrade before performing the post-upgrade tasks.

Hive Changes to ACID Properties

Hive 2.x and 3.x have transactional and non-transactional tables. Transactional tables have atomic, consistent, isolation, and durable (ACID) properties. In Hive 2.x, the initial version of ACID transaction processing is ACID v1. In Hive 3.x, the mature version of ACID is ACID v2, which is the default table type in HDP 3.0.

Native and Non-native Storage Formats in Hive

Storage formats are a factor in upgrade changes to table types. No change has occurred to storage formats in HDP 3.x, but storage formats determine table changes that occur during the upgrade. Hive 2.x and 3.x supports the following Hadoop native and non-native storage formats:

• Native: Tables with built-in support in Hive, such as those in the following file formats:

  • Text

  • Sequence File

  • RC File

  • AVRO File

- ORC File

- Parquet File

- Non-native: Tables that use a storage handler, such as the DruidStorageHandler or HBaseStorageHandler

HDP 3.x Upgrade Changes to Table Types

The following table compares Hive table types and ACID operations before an upgrade from HDP 2.x and after an upgrade to HDP 3.x. The ownership of the Hive table file is a factor in determining table types and ACID operations after the upgrade.

Table 4.1. HDP 2.x and 3.x Table Type Comparison

HDP 2.x				HDP 3.x	
Table Type	ACID v1	Format	Owner (user) of Hive Table File	Table Type	ACID v2
External	No	Native or non-native	hive or non-hive	External	No
Managed	Yes	ORC	hive or non-hive	Managed, updatable**	Yes
Managed	No	ORC	hive	Managed, updatable**	Yes
			non-hive	External, with data delete*	No
Managed	No	Native (but non-ORC)	hive	Managed, insert only**	Yes
			non-hive	External, with data delete*	No
Managed	No	Non-native	hive or non-hive	External, with data delete*	No

* See Dropping an External Table Along with the Data.

** Not SparkSQL-compatible

Optionally, install Data Analytics Studio

HDP 3.1.x does not include Hive View or Tez View. In lieu of these capabilities, users who upgrade from 2.6.x to 3.x can install Data Analytics Studio. Download Data Analytics Studio.

Other HDP 3.x Upgrade Changes

Managed, ACID tables that are not owned by the `hive` user remain managed tables after the upgrade, but `hive` becomes the owner.

After the upgrade, the format of a Hive table is the same as before the upgrade. For example, native or non-native tables remain native or non-native, respectively.

After the upgrade, the location of managed tables or partitions do not change under any one of the following conditions:

• The old table or partition directory was not in its default location `/apps/hive/warehouse` before the upgrade.

• The old table or partition is in a different file system than the new warehouse directory.

• The old table or partition directory is in a different encryption zone than the new warehouse directory.

Otherwise, the location of managed tables or partitions does change: The upgrade process moves managed files to `/warehouse/tablespace/managed/hive`. By default, Hive places any new external tables you create in HDP 3.x in `/warehouse/tablespace/external/hive`.

The `/apps/hive` directory, which is the former location of the Hive 2.x warehouse, might or might not exist in HDP 3.x.

Correcting Hive File Locations

To perform some steps in this procedure, you need to login as the HDFS superuser. If you use Kerberos, you need to become the superuser with a valid ticket.

If you had external files before the upgrade, the upgrade process carries the external files over to HDP 3.x with no change in location. The external files continue to reside in the `/apps/hive` directory. Check the `/apps/hive` directory for files that do not belong there after upgrading. Files that do not belong in `/apps/hive` are files that appear in the table above as managed files in HDP 3.x. The upgrade process should have moved the managed files to the new `/warehouse/tablespace/managed/hive` directory. Contacting Hortonworks Support for help to correct this problem is highly recommended. Alternatively, you can perform the following procedure to correct file locations of managed files:

1. Login as the HDFS superuser.

   ```
   $ sudo su – hdfs
   ```

2. Start Hive in Ambari 2.7.x.

3. On a node in your cluster, start Beeline in the background and a Hive shell in the foreground:

   ```
   $ hive
   ```

4. Change the location of the database and the table from the old location to the new location. For example:

   ```
   hive> ALTER DATABASE tpcds_bin_partitioned_orc_10
   SET LOCATION 'hdfs://ns1/apps/hive/warehouse/
   tpcds_bin_partitioned_orc_10.db';

   hive> ALTER TABLE tpcds_bin_partitioned_orc_10.store_sales
   SET LOCATION 'hdfs://ns1/apps/hive/warehouse/
   tpcds_bin_partitioned_orc_10.db/store_sales';
   ```

5. On the Hive Metastore node, log in as the HDFS superuser.

6. Set STACK_VERSION to the HDP version you are running. For example:

```
$ export STACK_VERSION=`hdp-select status hive-server2 | awk
'{ print $3; }'`
```

7. Run the following script:

```
$ /usr/hdp/$STACK_VERSION/hive/bin/hive --config /
etc/hive/conf --service strictmanagedmigration --
hiveconf hive.strict.managed.tables=true -m automatic --
modifyManagedTables --oldWarehouseRoot /apps/hive/warehouse
```

Add backticks to table references using dot notation

HDP 3.1.4 and later includes the Hive-16907 bug fix, which rejects `db.table` in SQL queries. The database name and the table name must be enclosed in backticks as follows: `db`.`table`; otherwise, Hive interprets the entire db.table string as the table name.

Updating HDFS Paths in Ranger Policies

You need to update the Hive warehouse path, which changes from `/apps/hive` in earlier releases to `/warehouse/tablespace/managed/hive/` in HDP 3.x. Apache Ranger policies that apply to tables in the new warehouse location do not work after the upgrade.

1. In Ambari, in **Services > Ranger > Summary**, click Ranger Admin UI.

2. Log into Ranger as the administrator.

3. In Service Manager, in HDFS, click the policy list name you want to update.

4. Click View for each policy, and edit the Resource Path of any policy that refers to the obsolete path.

Use the Hive Warehouse Connector to access Spark data

In HDP 3.x, you need to change the way you access Spark data as follows:

• Workflow change: You must use the Hive Warehouse Connector API to access any managed table in the Hive catalog from Spark.

• Service change: You must low-latency analytical processing (LLAP) in HiveServer Interactive to read ACID, or other Hive-managed tables, from Spark.

  You do not need LLAP to write to ACID, or other managed tables, from Spark. You do not need HWC to access external tables from Spark.

4.9. Upgrade Troubleshooting

In the event of a problem, contacting Hortonworks Support is highly recommended. Alternatively, you can perform these troubleshooting procedures.

Restoring the Hive Metastore

1. On the node where the database for Hive Metastore resides, create databases you want to restore. For example:

```
$ mysql -u hiveuser -p -e "create database
<hive_db_schema_name>;"
```

2. Restore each Metastore database from the dump you created. For example:

```
$ mysql -u hiveuser -p <hive_db_schema_name> < </path/to/
dump_file>
```

3. Reconfigure Hive Metastore if necessary. Reconfiguration might be necessary if the upgrade fails. Contacting Hortonworks Support for help with reconfiguration is recommended. Alternatively, in HDP 3.x, set key=value commands on the command line to configure Hive Metastore.

Solving Problems Using Spark and Hive

Use the Hive Warehouse Connector (HWC) and low-latency analytical processing (LLAP) to access Spark data after upgrading. HWC is a Spark library/plugin that is launched with the Spark app. HWC and LLAP are required for certain tasks, as shown in the following table:

Table 4.2. Spark Compatibility

Tasks	HWC Required	LLAP Required	Other Requirement/ Comments
Read Hive managed tables from Spark	Yes	Yes	Ranger ACLs enforced.
Write Hive managed tables from Spark	Yes	No	Ranger ACLs enforced.
Read Hive external tables from Spark	No	Only if HWC is used	Table must be defined in Spark catalog. Ranger ACLs not enforced.
Write Hive external tables from Spark	No	No	Ranger ACLs enforced.

Spark-submit and pyspark are supported. The spark thrift server is not supported.

Accessing Hive tables using SparkSQL

To access tables, which were converted to ACID tables during the upgrade, using SparkSQL, you create a new external table using Hive 3 and migrate the data from the managed to the new table.

1. Rename the managed table to `*_old`.

2. Migrate data from *_old to <new> external table using the original name in the historical or the default location (`/warehouse/tablespace/external/hive/<?>.db/<tablename>`).

```
CREATE EXTERNAL TABLE new_t as SELECT * FROM old_t;
```

Recovering missing Hive tables

1. Login as Superuser HDFS. For example:

```
$ sudo su - hdfs
```

2. Read the snapshots on HDFS that backup your table data. For example:

```
$ hdfs dfs -cat /apps/hive/warehouse/.snapshot/
s20181204-164645.898/students/000000_0
```

Example output for a trivial table having two rows and three columns might look something like this:

```
fred flintstone351.28
```

```
barney rubble322.32
```

3. In Hive, insert the data into the table if the schema exists in the Hive warehouse; otherwise, restore the Hive Metastore, which includes the schemas, from the database dump you created in the pre-upgrade process.

YARN Registry DNS instance fails to start

The YARN Registry DNS instance will fail to start if another process on the host is bound to port 53. Ensure no other services that are binding to port 53 are on the host where the YARN Registry DNS instance is deployed.

Class Loading Issue When Starting Solr

If you do not follow sequential steps during the upgrade, the Infra Solr instance may fail to start with the following exception:

```
null:org.apache.solr.common.SolrException: Error loading class
'org.apache.solr.security.InfraRuleBasedAuthorizationPlugin'
```

If you see this exception, follow the steps in this HCC article to work around the issue:

https://community.hortonworks.com/content/supportkb/210579/error-nullorgapachesolrcommonsolrexception-error-l.html

Ambari Metrics System (AMS) does not start

When the Ambari Metrics System (AMS) does not start after upgrade, you can observe the following log snippet in the HBase Master:

```
master.HMaster: hbase:meta,,1.1588230740 is NOT online; state={1588230740
 state=OPEN,
ts=1543610616273, server=regionserver1.domain.com,41213,1543389145213};
 ServerCrashProcedures=true.
Master startup cannot progress, in holding-pattern until region onlined
```

The workaround is to manually clean up the znode from ZooKeeper.

• If AMS mode = embedded, Remove the znode data from local filesystem path, e.g.:

```
 rm -f /var/lib/ambari-metrics-collector/hbase-tmp/zookeeper/zookeeper_0/
version-2/*
```

• If AMS mode = distributed, connect to the cluster zookeeper instance and delete the following node before restart:

```
 /usr/hdp/current/zookeeper-client/bin/zkCli.sh -server localhost:2181
[zk: localhost:2181(CONNECTED) 0] rmr /ams-hbase-unsecure/meta-region-server
```