Hortonworks Data Platform

Apache Ambari Upgrade Best Practices

(August 31, 2017)

docs.cloudera.com

Hortonworks Data Platform: Apache Ambari Upgrade Best Practices

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

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

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

Table of Contents

List of Tables

1. Overview

This guide helps you to plan, perform, and finalize your upgrade of Hortonworks Data Platform (HDP) using Apache Ambari.

If you have any questions or concerns, please contact Hortonworks Customer Support for guidance before upgrading your cluster.

2. Planning Your Upgrade

Follow the steps in this section to help plan your upgrade:

2.1. Developing Your Plan

Before you begin your upgrade, you should develop a plan for it, including the following tasks:

1. Identify and document the reasons for the upgrade.
2. Study the Ambari upgrade documentation to understand the process. See the Ambari Upgrade Guide.
3. Develop a written upgrade specification and procedure for your particular upgrade.
4. Complete the preupgrade verifications in the Completing the Preupgrade Verifications chapter of this guide.
5. Ensure that high availability is enabled for all components that require it. This is typically the namenode and secondary namenode, but can also include things such as access clients to applications and the NFS gateway.
6. Create a spreadsheet of tests that you might want to use to validate the health and stability of the cluster after the upgrade.
7. Simulate the upgrade on a virtual or development cluster so that you can practice upgrading in a test environment before you upgrade your production environment.

3. Completing the Preupgrade Verifications

After you complete the planning phase, you must verify your software versions, your environmental readiness, your application readiness, and the status of your HDP cluster.

Follow the steps in each of the following sections to complete the preupgrade verifications:

1. Verifying Versions [3]
2. Verifying Your Environment [3]
3. Verifying the Application [4]
4. Verifying the Status of the HDP Cluster [4]

3.1. Verifying Versions

Validate that the OS, Java Development Kit, and the metastore versions are compliant with the version of HDP and Ambari to which you plan to upgrade; for example, Oracle Database 12c is supported with Ambari 2.2.1.1. Confirm that your Ambari version is compatiable with your current version of HDP. See Determine Stack Compatibility in the *Automated Install with Ambari Guide*.

3.2. Verifying Your Environment

1. Validate that there are no broken symlinks to the configurations and that you are using the precise HDP version, from which you intend to upgrade. Be sure to check in the `/etc/hadoop/conf` directory.
2. Confirm that there is enough disk space in `/usr/hdp`, `/tmp`, and HDFS. The HDP 2.5.0 binaries require 6.5 GB of disk space. Do not underestimate your disk space requirements.
3. Clear the logs, especially if the logs are on a mount that is low on space and shared with the operating system.
4. Identify dead nodes and exclude or decommission them.
5. Remove any hosts in an abnormal state from the cluster. You can add them again after the upgrade.
6. Identify any special `*-env.sh` setting such as special bootstrap libraries. Document them in case you need to reapply them after the upgrade.
7. Identify corrupt blocks and either repair or make note of them.
8. Validate that you have access to each of the backing metastores used by HDP. These include, but are not limited to Ambari, Hive, Oozie, Ranger, and KMS. Each component has an upgrade process, some of which are fairly automatic. Ensure the connected user

to these databases have sufficient privileges to make schema changes as required by the upgrade.

9. Confirm that Ambari recognizes all the components that are installed in the cluster. If not, redeploy the component by using Ambari.

10. Ambari will not proceed with the upgrade if any components are in "maintenance" mode. Take time now to get those services up, healthy and out of maintenance mode.

3.3. Verifying the Application

1. Verify that any custom applications referencing `/usr/hdp/current...` are updated with new version. This might be found the `pom.xml` and could require that you recompile if there are changes.

2. Check your paths, which will change if you upgrade from versions before HDP 2.2.

3. Ensure that all libraries are referenced using `/usr/hdp/current/...`, to ensure that your script maintains its integrity for future upgrades.

4. Build processes, such as `maven` with declare library versions for the application in a `pom.xml` file must be adjusted, recompiled, and deployed during the upgrade cycle to ensure version parity with the cluster.

3.4. Verifying the Status of the HDP Cluster

1. Clean up old snapshots on HDFS to reclaim as much space before the upgrade as you can.

2. Run Balancer to ensure that data distribution is done.

3. Confirm that the current state of NN does not indicate that there is a previous upgrade still in progress.

4. Confirm that all the services indicate a healthy state.

5. Confirm that there are no open Ambari alerts.

6. Take note of any configuration differences between nodes.

7. Confirm that the schemas for Apache Oozie, Ambari, Hive Metastore, Ranger and Hue have been backed up.

8. If you intend to perform a multi-step upgrade process, or if you know that you will reach points during the upgrade that require significant effort and time, be sure to have additional backups of the various RDBMS metastores.

9. Confirm that the entire cluster is either quiesced or brought to a steady state.

4. Upgrading Your System

After you have fully planned an met all prequisites to your upgrade, follow the steps in the Ambari Upgrade Guide.

5. Completing the Postupgrade Verifications

After performing your upgrade, you should take some steps to ensure that your cluster meets all of your requirements. This verification involves reconfigurating manually, updating metastore schema, validating component behavior, and finalizing the upgrade.

Follow the steps in each of these sections to complete the postupgrade verifications:

1. Reconfiguring Manually [6]
2. Upgrading the Metastore Schema [6]
3. Validating the Upgrade Manually [6]
4. Finalizing the Upgrade [7]

5.1. Reconfiguring Manually

1. Based on Preupgrade Discovery, redeploy applications with the new HDP compiled libraries.
2. Recompile your framework, if there is custom code that is dependent on the HDP APIs, for example, MapReduce, HCat, Slider, HBase or Storm.
3. Configure clients to use the correct shared libraries.

5.2. Upgrading the Metastore Schema

Ensure that the metastore JDBC drivers are upgraded to the latest version available.

5.3. Validating the Upgrade Manually

Although the upgrade process includes some basic service checks, those checks are cursory. You must test the critical components that you identified during your upgrade planning and thereby validate the entire cluster.

Table 5.1. Validating the Upgrade Manually by Component

Component	Validation Instruction
Core Hadoop	See Validating the Core Hadoop Installation
Apache HBase	See Validating the HBase Installation
Apache Phoenix	See Validating the Phoenix Installation
Apache Tez	See Validating the Tez Installation
Apache Hive and Apache HCatalog	See Validating Hive-on-Tez Installation
Apache Pig	See Validating the Pig Installation
Apache WebHCat	See Validating the WebHCat Installation

Component	Validation Instruction
Apache Oozie	See Validating the Oozie Installation
Apache Ranger	See Validating the Ranger Installation
Hue	See Validating the Hue Installation
Apache Sqoop	See Validating the Sqoop Installation
Apache Mahout	See Validating the Mahout Installation
Apache Storm	See Validating the Storm Installation
Apache Spark	See Validating the Spark Installation
Apache Kafka	See Validating the Kafka Installation
Apache Zeppelin	See Validating the Zeppelin Installation
Apache Accumulo	See Validating the Accumulo Installation
Apache Falcon	See Validating the Falcon Installation
Apache Knox	See Validating the Knox Installation
Apache Slider	See Validating the Slider Installation

Your preupgrade planning process should identify a list of critical applications, streaming applications, Hive jobs, and HBase client access points. Have a test plan put together ensures your components are working properly after the upgrade. This is especially important for applications that need to be re-compiled.

You should have ready a series of tests that test:

1. MapReduce

   Consider running Teragen or Sort on a substantial dataset. Use these to baseline the new configuration for future reference.

2. HBase

3. Hive with both engines: MR and Tez

4. Storm

5. Kafka

6. Other cluster-utilitzed components

5.4. Finalizing the Upgrade

After you test the functional aspects of the cluster, you must finalize the cluster upgrade, which is otherwise left in upgrade start status state. Major changes to the file system HDFS are not complete until the process is finalized, and the system cannot reclaim space from any clean up efforts to follow. You cannot run the balancer, and major HBase compactions will consume massive amounts of space. Even smaller cluster upgrades should be finalized promptly.

If you had remove unhealthy hosts from the cluster before the upgrade, you can now add them back, as if they were new additions. You cannot run the balancer, and major HBase compactions will consume massive amounts of space. Even smaller cluster upgrades should be finalized promptly.